WHY SAMPLE WHAT YOU CAN ENUMERATE? EXACT POLICY OPTIMIZATION FOR GENOMIC TOOL SELECTION

Anonymous authors
Paper under double-blind review

ABSTRACT

Reinforcement learning over a frozen reasoner has become a common recipe for teaching a policy which tools to invoke. We show that this recipe becomes structurally mismatched in specialist scientific settings where the complete tool-subset space is enumerable. There, a few recurring capabilities cover the domain, so the subset space is combinatorial yet small enough to enumerate, and GRPO still estimates an action expectation from a handful of sampled rollouts. Worse, the approximation degrades as the policy concentrates: it resamples its preferred subsets, sampled rewards collide, and the group-normalized advantage vanishes. On genomic reasoning the closed-form fraction of questions yielding no reward signal rises from 0.2% under a uniform reference policy to 20.8% after GRPO training. As a remedy, we introduce FGPO (Full-Group Policy Optimization), which (1) scores every tool subset and optimizes the exact action expectation, and (2) precomputes the reward of each question–subset pair into an exhaustive table, removing reasoner calls from training. Across five frozen reasoners and three genomic benchmarks, FGPO outperforms GRPO in all 15 settings by 6.75 points on average and up to 14.20, while a standard on-demand GRPO schedule would require $2.4\times$ as many frozen-reasoner reward evaluations and, on GenomeQA, FGPO cuts invoked tools per question from 2.36 to 1.40. With the same table, it beats supervised routing and matches or beats reward regression in every setting.

1 INTRODUCTION

Large language models can interpret specialized scientific questions but cannot reliably perform the precise computations many of them require. Genomic reasoning is a representative case: an LLM understands promoters, transcription-factor binding and splice sites, yet answering such questions demands explicit computation over raw nucleotide sequences: motif scanning, splice-site scoring, composition analysis (Jin et al., 2024). External tools supply exactly these capabilities, and a productive line of work therefore trains a policy to select which tools a frozen reasoner should receive, optimizing that policy with reinforcement learning: VisTA for visual tools (Huang et al., 2025), AuTAgent for audio tools (Tong et al., 2026), and reward-shaped variants (Qian et al., 2026; Jin et al., 2025), almost always with GRPO (Shao et al., 2024).

Despite its success elsewhere, this recipe carries an unavoidable drawback in the specialist regime studied here. Because a domain reuses a small set of recurring capabilities, a compact library covers it; and even where the global registry is large, the per-query active set is small: on BFCL, an adaptive shortlist of about seven tools (7.4 ± 2.5) from a 370-tool registry retains 90.3% correct-tool coverage, nearly matching 90.8% with fifty (Repantis et al., 2026). Our four genomic tools admit only $2^4{=}16$ subsets, making the action expectation exactly computable, yet GRPO still approximates it through sampled rollouts. Even at this scale selection matters: as illustrated in Figure 1a, the frozen reasoner scores 39.28% without tools while a per-question oracle over the same library reaches 77.41%, leaving 38.1 points of recoverable headroom that invoking all tools (49.39%) does not capture. But sampling that space is not merely wasteful: it degrades precisely as the policy concentrates. A group teaches the policy nothing when its sampled rewards coincide, since every normalized advantage is then zero, and a concentrating policy resamples the same subsets. Figure 1b shows this *dead-group* rate, computed in closed form, climbing from 0.2% under a uniform reference policy to 20.8% after GRPO

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

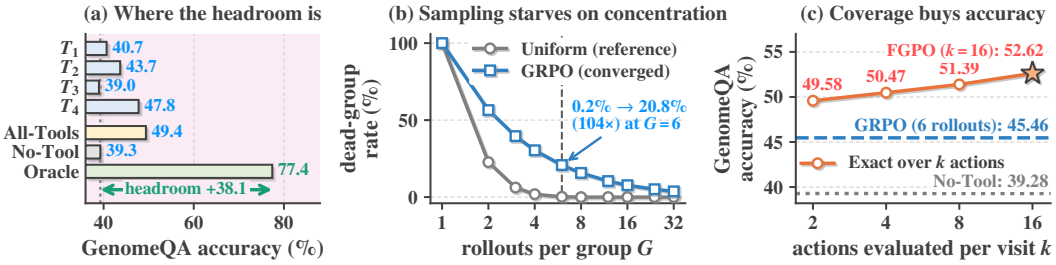


Figure 1: **Why sample when enumeration is possible?** (a) Individual tools, All-Tools, No-Tool, and a per-question oracle on the 3,590 GenomeQA test questions; the oracle reveals 38.1 points of recoverable headroom. (b) Closed-form probability that all G rollouts receive identical reward, forming a *dead* group with zero reward advantage. (c) Accuracy when the same training construction observes only k uniformly sampled actions per visit.

training, reaching 79.6% under the differential reward GRPO trains with; Figure 1c shows accuracy rising monotonically as the optimizer is shown more of the action space. This motivates the central question of this paper: *Why sample an expectation that can be computed exactly?*

As a remedy, we introduce FGPO (Full-Group Policy Optimization), an exact policy optimization framework for enumerable tool-selection spaces. FGPO incorporates (1) an exact objective that scores every tool subset and optimizes the complete action expectation, so each update sees the whole action space rather than a sample of it, and (2) an exhaustive reward table that precomputes each question–subset pair once, removing frozen-reasoner calls from the training loop entirely. For autoregressive LLM policies we further use per-token length-normalized candidate scoring and entropy regularization to obtain a practical realization of this objective.

Our contributions are summarized as follows:

- We formulate tool-augmented genomic reasoning as query-dependent combinatorial tool-subset selection and reveal 38.1 points of per-question oracle headroom over the unaided reasoner, demonstrating the importance of selecting specialist capabilities correctly.
- We first show that sampled policy optimization is structurally mismatched to enumerable tool-selection spaces. On GenomeQA, the closed-form dead-group rate rises from 0.2% to 20.8% after GRPO training, with similar rates of 17.4–17.8% on two additional benchmarks; controlled k-subset experiments further show that accuracy improves monotonically as the same training construction is given broader uniform action coverage.
- We introduce FGPO, which replaces sampled optimization with exact optimization over all enumerable tool subsets. Across five reasoners and three genomic benchmarks, FGPO outperforms GRPO in all 15 settings by 6.75 points on average and up to 14.20 points, while a standard on-demand GRPO schedule would require 2.4× more frozen-reasoner reward evaluations and on GenomeQA FGPO reduces average invoked tools from 2.36 to 1.40.

2 FGPO: Exact Policy Optimization over Enumerable Tool Subsets

This section develops FGPO as exact policy optimization for LLM tool selection: the objective over all 2^n tool subsets (Section 2.2), two choices its LLM instantiation needs plus a controlled candidate-scoring evaluation (Section 2.3), and an exhaustive reward table that precomputes $r(s, a)$ once (Section 2.4). Algorithm 1 in Appendix A states the whole procedure.

2.1 Setting

Let $\mathcal{D}$ denote the training-question distribution. A frozen reasoner R answers multiple-choice genomic questions $s \sim \mathcal{D}$, optionally given the output of a subset $a \subseteq \{T_1, \ldots, T_n\}$ of tools run on the question's sequence. The tools are independent analyses of that sequence, none consuming another's output, so a subset fully specifies the execution. A policy $q_\theta(a \mid s)$, with θ the parameters of a LoRA adapter, selects the subset; the action space is $\mathcal{A} = 2^{\{T_1,\ldots,T_n\}}$ with $|\mathcal{A}| = 2^n$ (n=4 in all main

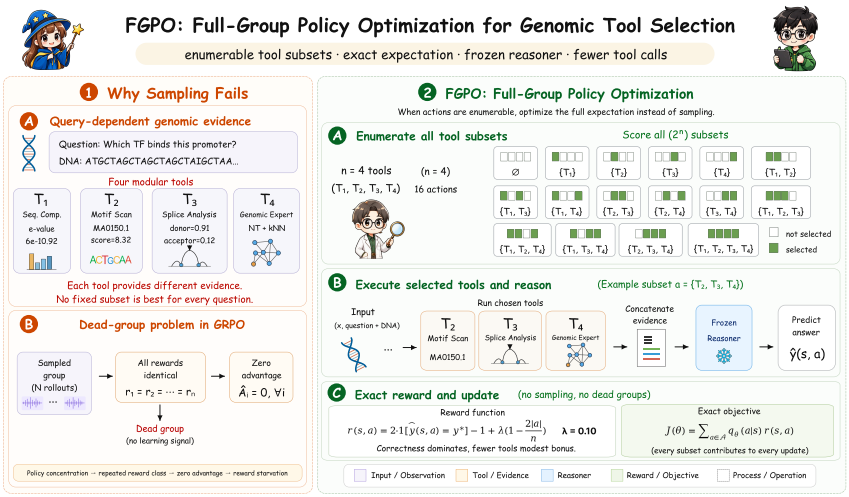


Figure 2: **Overview of FGPO.** For each question, FGPO enumerates all 2^n tool subsets and precomputes their rewards once using the frozen tools and reasoner. Training then scores every subset and optimizes the exact action expectation, so every action contributes to every update without action sampling. In contrast, GRPO draws only G sampled rollouts; when their rewards coincide, all group-normalized advantages vanish, producing a *dead group*.

experiments, 16 actions; Table 1 lists the library). With $y^\star$ the gold option and $\hat{y}(s,a)$ the reasoner's answer given the evidence of subset a, the reward scores correctness and adds a parsimony tie-break an order of magnitude smaller:

$$r(s,a) \;=\; \underbrace{2\cdot\mathbf{1}[\hat{y}(s,a)=y^\star]-1}_{\pm 1} + \lambda\left(1-\frac{2|a|}{n}\right), \qquad \lambda = 0.10, \tag{1}$$

so correctness always dominates and, among subsets that agree on it, fewer tools score higher. We follow the *setting* of Tong et al. (2026) (frozen reasoner, tool-subset action space, RL-trained selector) but not its differential reward (Section 3.1). The policy is a 7B LLM with a LoRA adapter (Hu et al., 2021) that emits the subset as a short indexed string over *anonymous* tool slots, with no tool names or descriptions, so any routing it learns comes from reward rather than text (Appendix G).

Table 1: **The modular tool library.** All tools are frozen; T_4 uses source-training kNN. Solo worth is shown in Figure 1a.

ID	Tool Module	Source
T_1	Seq. Composition	Harris et al. (2020)
T_2	Motif Scan	Raluseviciute et al. (2024)
T_3	Splice Analysis	Yeo & Burge (2003)
T_4	Genomic Expert	Dalla-Torre et al. (2025)

2.2 The exact objective

With a frozen, greedily decoded reasoner we treat $r(s,a)$ as effectively deterministic (an independent live pipeline agrees with the cache within 0.05 points, Appendix C). The *action* expectation in the policy-gradient objective and its gradient is then a finite sum, evaluated exactly; the expectation over questions is minibatched as usual,

$$J(\theta) \;=\; \mathbb{E}_{s\sim\mathcal{D}} \sum_{a\in\mathcal{A}} q_\theta(a\,|\,s)\, r(s,a), \qquad \nabla_\theta J \;=\; \mathbb{E}_s \sum_{a\in\mathcal{A}} r(s,a)\,\nabla_\theta\, q_\theta(a\,|\,s), \tag{2}$$

computable without sampling whenever 2^n candidate evaluations per visit are affordable. FGPO optimizes Eq. 2 directly over the tool-subset space, connecting to expected and all-action policy gradients (Ciosek & Whiteson, 2020; Asadi et al., 2017) in the contextual-bandit setting (Langford & Zhang, 2007). Contrast GRPO (Shao et al., 2024), which samples G rollouts from the autoregressive generation policy. Let $p_\theta^{\text{gen}}(a\,|\,s)$ denote the categorical distribution over parsed tool subsets induced

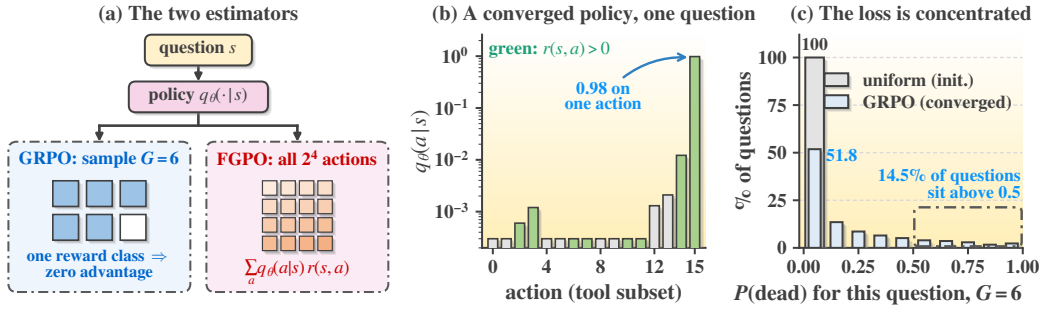


Figure 3: **What each estimator sees.** (a) The two estimators on the same state. (b) The converged GRPO policy's exported distribution over the 16 subsets for one GenomeQA question; green marks positive reward. (c) Per-question dead probability at G=6 over all 3,590 test questions, with reward classes from Eq. 1; the mean is 20.8%, as reported in Section 3.4.

by that sampler; then $a_1, \ldots, a_G \sim p_\theta^{\text{gen}}(\cdot \mid s)$, and GRPO steps on group-normalized advantages, where r is whatever reward that arm trains on, the differential reward in our experiments (Section 3.1):

$$\hat{A}_i = \frac{r(s,a_i) - \mu}{\sigma + \epsilon}, \qquad \mu = \frac{1}{G}\sum_{j=1}^{G} r(s,a_j), \qquad \sigma^2 = \frac{1}{G}\sum_{j=1}^{G}\big(r(s,a_j) - \mu\big)^2. \tag{3}$$

with $\epsilon > 0$ for numerical stability. Two limitations follow. Whenever the G drawn rewards coincide every numerator is zero, so all advantages vanish regardless of ϵ, a *dead* group, which Section 3.4 shows reaches 104×–178× the uniform-reference rate. And because the group is drawn from p_θ^{gen} itself, coverage shrinks exactly as the sampling policy concentrates, which is what optimization produces. Eq. 2 evaluates every subset without sampled groups: concentration can shrink reward gradients through q_θ, but no action is omitted by sampling (Appendix F.1).

Figure 3 makes this concrete on real data: the converged GRPO policy puts 98% of its mass on a single subset (panel b), and the per-question dead probability is heavily skewed (panel c), so sampling goes blind precisely where the policy has already committed.

2.3 Instantiating the exact estimator on an LLM policy

Two choices separate Eq. 2 from a working method on an LLM policy, each motivated by an observed failure; we additionally use a controlled candidate-scoring evaluation to compare trained policies under an identical decoding rule.

(i) Per-token length normalization. The 2^n actions are strings of 5–13 tokens and the empty subset is the shortest, so a softmax over raw sequence log-probabilities embeds a prior toward invoking nothing, a bias absent when actions are abstract indices. With y_a the token string encoding a and π_θ the policy's next-token distribution, we score candidates by their per-token mean. Here q_θ denotes the candidate-normalized categorical distribution used by FGPO ("exact" refers to the full sum over these candidates, not to marginalizing free-form generation); it is distinct from the generation-induced p_θ^{gen} used by the GRPO sampler in Eq. 3:

$$\ell_\theta(a \mid s) = \frac{1}{|y_a|}\sum_{t=1}^{|y_a|} \log \pi_\theta(y_{a,t} \mid s,\, y_{a,<t}), \qquad q_\theta(a \mid s) = \frac{\exp \ell_\theta(a \mid s)}{\sum_{a' \in \mathcal{A}} \exp \ell_\theta(a' \mid s)}. \tag{4}$$

(ii) Entropy regularization. To preserve per-question discrimination we optimize

$$J_\beta(\theta) = J(\theta) + \beta\, \mathbb{E}_{s\sim\mathcal{D}}\, H(q_\theta(\cdot \mid s)), \qquad H(q) = -\sum_{a\in\mathcal{A}} q(a) \log q(a), \qquad \beta = 0.03, \tag{5}$$

which encourages broader candidate probabilities during optimization.

(iii) A controlled candidate-scoring evaluation. Training already scores every candidate, so a controlled evaluation can reuse that score rather than generating free-form text:

$$\hat{a}(s) = \arg\max_{a\in\mathcal{A}}\ \ell_\theta(a \mid s). \tag{6}$$

Format failure is removed by construction rather than by parser engineering. The rule is available to any autoregressive policy, so we apply it to the baselines too: scoring all 16 candidates instead of generating freely moves GRPO $45.46 \rightarrow 45.13$, SFT $42.17 \rightarrow 39.30$ and FGPO $52.62 \rightarrow 51.84$, leaving FGPO ahead by 6.71 points under an inference rule identical across arms. Table 2 reports free generation throughout.

2.4 The exhaustive reward table

We precompute $r(s, a)$ for all 2^n actions of every training question. The table is memoization (FGPO could evaluate rewards on demand) but it makes the economics explicit: building the table takes $2^n\,|\mathcal{D}| = 16 \times 2{,}002 = 32{,}032$ reasoner calls, whereas the reported GRPO schedule (200 steps $\times$ 384 rollouts) entails 76,800 reward evaluations, a $2.4\times$ difference for a standard on-demand implementation. In our controlled experiments both arms read the same cached outcomes, so this cost is an accounting fact about the schedules rather than a difference between the runs we report; and everything downstream becomes a lookup with no further frozen-reasoner calls. Under a constrained budget, evaluating a uniformly drawn k-subset of actions with q_θ renormalized on it degrades gracefully: at k=2, one third of GRPO's reward evaluations per visit, accuracy still leads by 4 points (Section 3.5).

3 Experiments

We organize the evaluation around five questions, each answered by the correspondingly numbered subsection. **Q1:** Does FGPO outperform sampled and offline baselines under matched data, prompts and adapters, and do the learned selections transfer across frozen reasoners? **Q2:** What selection behaviour does each objective actually produce? **Q3:** *Why* does the sampled estimator underperform, and does the mechanism replicate? **Q4:** Does action-space coverage drive the gain? **Q5:** Given the exhaustive table, does the training objective matter? Additional numerical results, including the full transfer grid, paired significance tests, ceiling analyses, dead-group grids, library ablations, GRPO/DPO sweeps and cost accounting, are in Appendices C–F.

3.1 Experiment Setup

Benchmarks and tools. Three multiple-choice genomic QA suites on open corpora: *GenomeQA* (Long et al., 2026) (3,590 test / 492 dev) and two held-out cross-domain suites, *GenBench-X* and *BM4* (1,000 each) (Grešová et al., 2023; De Almeida et al., 2022). The 2,002 training questions come from the Nucleotide Transformer downstream tasks (Dalla-Torre et al., 2025); the selector is trained on none of the three evaluation benchmarks, and T_4 draws its kNN references only from training partitions, never from test sequences or labels. The n=4 tools span sequence composition (T_1), motif scanning (T_2), splice-site analysis (T_3), and a kNN predictor over frozen Nucleotide-Transformer embeddings (T_4). **Policy, baselines and reasoners.** The policy is Qwen2.5-7B-Instruct (Qwen et al., 2025) with a rank-16 LoRA; GRPO, DPO and SFT use the official TRL implementations (von Werra et al., 2020) on identical prompts, data and adapters, SFT reported at its development-selected checkpoint, GRPO at its best checkpoint and DPO at the best cell of its sweep, with DPO given the *true* argmax from the exhaustive table as its chosen response. The five frozen reasoners are Qwen2.5-1.5B/7B (Qwen et al., 2025), Qwen3-8B (Yang et al., 2025), Mistral-7B (Jiang et al., 2023) and InternLM2.5-7B (Cai et al., 2024). Significance uses question-paired McNemar tests (McNemar, 1947; Dietterich, 1998). Appendix B specifies all of these exactly, including the three training-free references *Random*, *All-Tools* and *Tools w/ Desc*.

3.2 A1: FGPO outperforms sampled and offline baselines

Table 2 reports every selection strategy under matched conditions. **FGPO is the best or tied-best method in** 14 **of the** 15 **cells**, exceeding GRPO by 6.75 points on average. More importantly, FGPO outperforms GRPO in all 15 benchmark–reasoner cells, showing that the gain persists across frozen reasoners rather than being tied to the Qwen3-8B reasoner that supplied the training rewards. On GenomeQA FGPO also calls fewer tools than GRPO (1.40 vs. 2.36 per question) and uses 3.5–$4.1\times$ fewer input tokens than exhaustive All-Tools (Table 8, Appendix C). The gain is not an inference-rule artifact: with every arm scoring all 16 candidates, FGPO leads GRPO by 6.71 (Section 2.3). Nor is it

Table 2: **Main comparison.** Accuracies (%). Qwen3-8B supplied the training rewards; the other four reasoners receive the same policy without adaptation. Trained rows share data, prompts and adapters; GRPO keeps Tong et al.'s differential reward while FGPO optimizes Eq. 1 (Appendix D.4 examines the parsimony term). Per row, the top three are in **red**, **green** and **blue**; ties share a colour. ‡Trained on Eq. 1, the reward FGPO optimizes: GRPO with its sampled estimator and DAPO with dynamic sampling and clip-higher.

Benchmark	Reasoner	No-Tool	Random	All-Tools	Tools w/ Desc	SFT	DPO	GRPO	GRPO‡	DAPO‡	FGPO
GenomeQA	Qwen3-8B†	39.28	45.38	**49.39**	**47.74**	42.17	39.33	45.46	39.25	39.16	**52.62**
	Qwen2.5-1.5B	38.25	40.78	**42.92**	**44.07**	40.72	38.33	42.87	38.13	38.25	**43.79**
	Qwen2.5-7B	37.86	44.93	**48.91**	**47.10**	40.25	38.22	46.74	38.02	37.88	**50.58**
	Mistral-7B	37.83	34.82	23.09	**41.00**	**38.08**	37.69	35.68	37.69	37.74	**42.90**
	InternLM2.5-7B	38.36	45.38	47.94	**48.66**	42.48	38.77	**48.05**	38.50	38.47	**52.53**
GenBench-X	Qwen3-8B†	41.20	54.20	**58.90**	**59.00**	46.60	41.30	51.90	41.20	41.20	**65.40**
	Qwen2.5-1.5B	41.40	50.50	**58.50**	**54.20**	42.00	41.40	48.80	41.40	41.40	**60.00**
	Qwen2.5-7B	42.10	57.30	**66.60**	**59.90**	51.50	42.30	57.50	42.10	41.90	**68.90**
	Mistral-7B	40.30	43.50	**45.40**	**52.00**	41.30	40.30	44.00	40.40	40.30	**58.20**
	InternLM2.5-7B	39.70	54.20	**63.10**	**59.70**	50.00	39.90	54.10	39.70	39.70	**66.90**
BM4	Qwen3-8B†	43.20	48.00	**53.30**	47.30	43.50	43.10	**52.30**	43.00	43.20	**54.80**
	Qwen2.5-1.5B	42.60	47.00	**50.50**	47.30	42.70	42.70	**48.80**	42.60	42.50	**51.80**
	Qwen2.5-7B	43.60	48.00	**55.00**	47.50	44.30	43.50	**52.90**	43.40	43.20	**55.30**
	Mistral-7B	42.30	**43.70**	42.80	**48.90**	41.70	42.30	**45.20**	42.50	42.10	**48.90**
	InternLM2.5-7B	42.60	48.90	**53.70**	47.10	45.20	42.40	**51.70**	42.50	42.30	**54.70**

†Training reasoner; the other four columns transplant the same policy without adaptation.

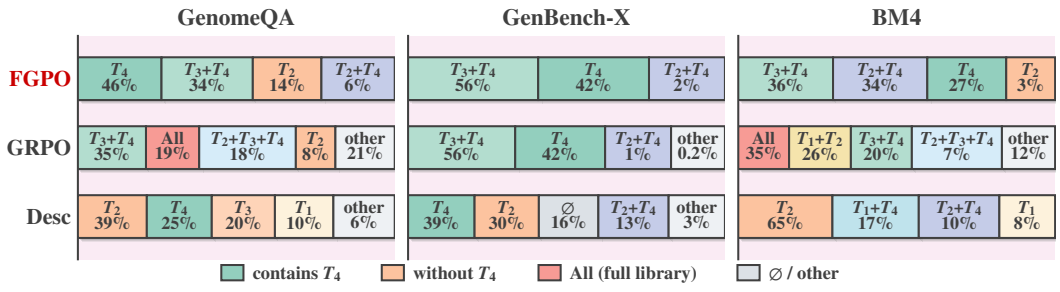


Figure 4: **What each selector actually calls.** Top subset combinations per benchmark. Teal/blue marks subsets containing T_4, orange/yellow those without, red the full library, and gray the empty set or residual subsets. Widths are compressed for legibility; labels give the displayed shares. FGPO concentrates on a few T_4-containing subsets, whereas GRPO spreads broadly, including *All* on 19% of GenomeQA and 35% of BM4.

the reward: trained on Eq. 1 (GRPO‡, DAPO‡), GRPO and DAPO collapse onto the no-tool policy, within 0.4 points of No-Tool in every cell, with closed-form dead-group rates of 92.9% and 97.7% at G=6 against 20.8% for the reported GRPO (Appendix E.3).

What the selections look like question by question. On the 418 questions where FGPO is right and GRPO is wrong, the selected tool typically returns a per-option *contrast* rather than a decisive reading of the winning option alone (Figure 6c; Appendix C works through three case studies).

3.3 A2: FGPO learns query-dependent genomic evidence routing

Each baseline fails in its own shape. Read in aggregate (Figure 4), on GenomeQA the prompted selector calls exactly one tool on 93.6% of questions, selecting a singleton subset on nearly every question, while GRPO calls two or more on 81% and spends three or four tools on 37.7%. FGPO does neither: it never exceeds two tools, drops T_1 entirely (0.0% on all three benchmarks, against GRPO's 21–64%) while concentrating on T_4, the tool that uniquely rescues the most questions (308), calling it on 86.4% of GenomeQA questions and at least 96% on the other two. Dropping T_1 is a

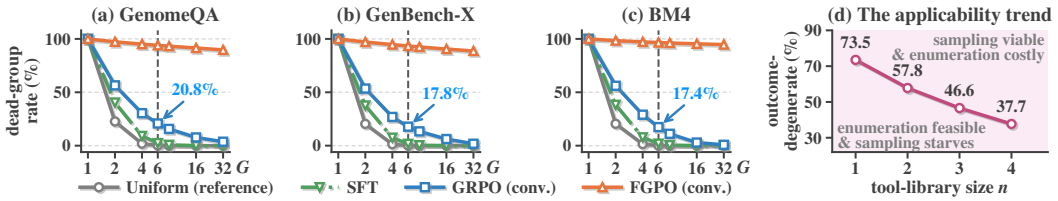


Figure 5: **Dead-group rate replicates across benchmarks.** (a)–(c) Closed-form rate vs. group size G; dashed line marks GRPO's G=6. Panel (a) adds the offline and exact arms to Figure 1b; (b),(c) repeat it on the held-out benchmarks. (d) *Outcome-degenerate* fraction (all subsets induce the same correctness outcome) vs. library size, averaged over sub-libraries.

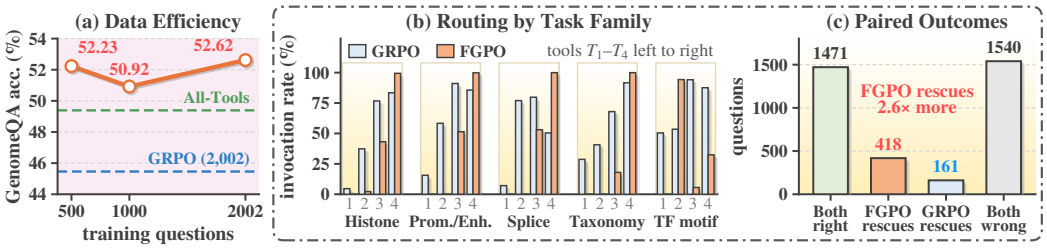


Figure 6: **Additional analysis.** (a) 500 training questions already beat GRPO trained on 2,002 questions and the training-free All-Tools baseline. (b) Per-tool invocation rate *within* each task family (Figure 4 aggregates over them). GRPO calls every tool at a similar rate whatever the question type; FGPO routes (T_2 for TF motif, T_4 elsewhere) and never calls T_1 on any benchmark, against GRPO's 21–64% across the three: a learned refusal. (c) FGPO rescues 2.6× as many questions as GRPO.

decision, not a free lunch: removing it from the library costs the per-question oracle 2.1–3.3 points across the three benchmarks (Table 13), so FGPO is giving up reachable questions in exchange for never paying T_1's cost on the rest. None of this was supervised (tool slots are anonymous), so the shape of the policy is a statement about what the reward could be made to reveal.

Offline objectives collapse onto the marginal mode. On the GenomeQA test set the empty subset is reward-optimal on 61.9% of questions (the reasoner is often right unaided, and the parsimony term then favors calling nothing). SFT clones this marginal: its argmax is the empty set on 92.8% of questions. DPO collapses entirely: across three β values and seven checkpoints it selects 0.00–0.10 tools and lands on the no-tool floor (39.33 vs. 39.28), because 58.1% of chosen responses are the empty string and a pairwise ranking objective is satisfied by the mode (Tong et al., 2026). **GRPO does not collapse; it starves**, which is Q3.

3.4 A3: Sampling starves *as* the policy concentrates

For the closed-form diagnostic, let $p_\theta^{\text{diag}}(a \mid s)$ denote p_θ^{gen} conditioned on the 16 valid subset strings (each scored by teacher forcing and renormalized over the 16; Appendix F.2). The probability that G independent draws all land in one reward class, a dead group, is closed-form: $P_{\text{dead}}(s) = \sum_v \left(\sum_{a:r(s,a)=v} p_\theta^{\text{diag}}(a|s) \right)^G$. At GRPO's G=6: 0.2% of questions are dead under a uniform reference policy, 20.8% under the converged GRPO policy (104×), replicated at 17.8%/17.4% (178×/174×) on GenBench-X/BM4 (Figure 5). These rates use the reward classes of Eq. 1, the finest any arm induces. Under the coarser differential reward GRPO actually trains on, the dead fraction reaches 79.6% at convergence, so these figures are conservative.

The starvation is not benign. A vanishing signal would be unremarkable if it dried up only where the policy had already succeeded. It does not: dead-group probability is near-independent of correctness (54% of the dead mass sits on wrong answers), and 6.7% of all questions are simultaneously dead, wrong, and repairable by some subset in the policy's own action space. The policy stops learning on them before it solves them, while the exact estimator's signal on them never vanishes by sampling.

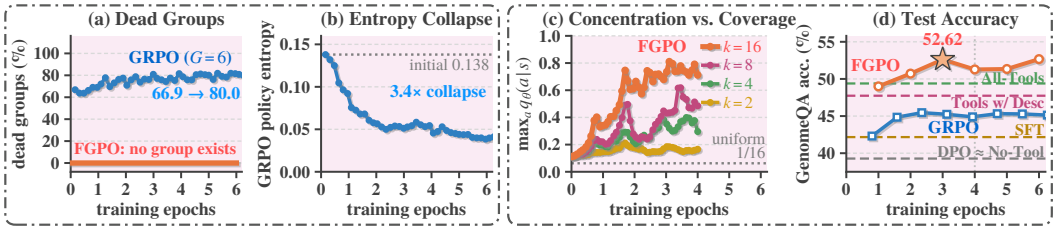


Figure 7: **The mechanism, measured live during training.** (a) The observed dead-group rate rises throughout GRPO training, while the exact objective has no sampled groups by construction. (b) GRPO's policy entropy decreases in parallel. (c) FGPO also becomes highly concentrated while retaining complete action coverage. (d) Test accuracy against the reference lines from Table 2: FGPO surpasses all of them, whereas GRPO stays below All-Tools and Tools w/ Desc.

Live telemetry under the coarser differential reward actually used by GRPO shows the same dynamic during temperature-1.0 generation; malformed completions additionally join the same reward class. Over the reported GRPO run the observed dead-group fraction rises from 66.9% to 80.0% (Figure 7a) as its policy entropy collapses by $3.4\times$ (Figure 7b). These observations are consistent with the mechanism formalized in Appendix F.1: greater concentration of probability mass over reward classes increases the probability of a dead group. FGPO also concentrates, peaking near 0.8 on its top action against a uniform $1/16$ (Figure 7c), while avoiding sampled dead groups. GRPO's test accuracy (Figure 7d) peaks at 37.5% of training and is flat thereafter while its training reward gains a further 21%, so the objective keeps improving after it has stopped buying accuracy, and it never reaches the two strongest training-free references of Table 2 (All-Tools, Tools w/ Desc); FGPO clears both.

The same tables give a library-size trend. Call a question *outcome-degenerate* if all 2^n subsets induce the same correctness outcome, so tool selection cannot alter whether the frozen reasoner is right. Averaging over all $\binom{4}{n}$ sub-libraries of each size, Figure 5d shows this fraction falling from 73.5% at one tool to 37.7% at four. Larger libraries thus expose more outcome-diverse actions at the same time that exhaustive enumeration becomes more expensive. The two estimators' applicability regions are complementary: this paper characterizes the regime where exact optimization is feasible and sampled optimization is most information-starved.

3.5 A4: A controlled coverage intervention: the k-subset dose–response

The dead-group analysis is observational; we now test it with an intervention. We hold the data, ordering, hyperparameters and action-scoring construction fixed and vary only the number of candidate actions the estimator is shown: at each visit it sees a uniformly drawn subset of k of them, with q_θ renormalized on that subset, so each k is its own objective and k=16 recovers FGPO. Policy-model compute per step is unchanged, since all candidate strings are scored in every arm. Accuracy at the (dev-selected) peak epoch is strictly monotone in k: $49.58 \to 50.47 \to 51.39 \to 52.62$ for $k = 2, 4, 8, 16$ (paired McNemar: k=16 over k=2, $p < 10^{-4}$; over k=8, $p = 0.011$; Figure 1c and Table 3). Two readings follow. First, what the estimator sees of the action space is what the method learns, the controlled counterpart of Section 3.4. Second, even k=2 (49.58) exceeds six-rollout GRPO (45.46): uniform draws do not concentrate with the policy, consistent with GRPO's deficit having two components, coverage *and* sampling from a concentrated distribution, though only coverage is varied here. The k dial also spans a coverage–budget continuum: with two reward evaluations per visit, one third of GRPO's, the budgeted variant still leads by 4 points.

Table 3: **The coverage dial.** Accuracy by epoch when the objective sees only k uniformly drawn actions per visit. Everything else held fixed. Top three per column in **red**, **green**, **blue**.

k	ep1	ep2	ep3	ep4	peak
2	**47.55**	49.14	49.58	**49.44**	49.58
4	47.27	**50.25**	**50.47**	45.24	**50.47**
8	**49.14**	**50.89**	**51.39**	**50.56**	**51.39**
16 (FGPO)	**49.00**	**50.72**	**52.62**	**51.28**	**52.62**
GRPO (G=6)		—			45.46

Table 4: **Objective ablation.** Only the loss differs: same backbone, adapter, data, dev selection; all arms score all 16 candidates. Accuracy (%); row best in **red**, second in **green**. †Training reasoner.

Benchmark	Reasoner	Hard-label CE	Soft-label CE	Regression	FGPO	Δ vs. regression
GenomeQA	Qwen3-8B†	39.30	46.27	**46.55**	**51.84**	+5.29
	Qwen2.5-1.5B	38.44	40.61	**41.73**	**43.65**	+1.92
	Qwen2.5-7B	38.83	45.24	**45.82**	**50.47**	+4.65
	Mistral-7B	37.52	**39.86**	37.13	**42.48**	+5.35
	InternLM2.5-7B	39.69	45.43	**47.38**	**52.62**	+5.24
GenBench-X	Qwen3-8B†	42.80	46.20	**62.60**	**65.40**	+2.80
	Qwen2.5-1.5B	41.50	44.10	**57.00**	**60.10**	+3.10
	Qwen2.5-7B	43.80	47.20	**63.30**	**68.70**	+5.40
	Mistral-7B	41.00	41.90	**53.60**	**57.90**	+4.30
	InternLM2.5-7B	41.70	44.20	**62.30**	**66.80**	+4.50
BM4	Qwen3-8B†	43.10	47.50	**55.60**	**54.50**	−1.10
	Qwen2.5-1.5B	42.60	47.40	**50.10**	**51.10**	+1.00
	Qwen2.5-7B	44.40	49.30	**53.10**	**55.00**	+1.90
	Mistral-7B	42.40	47.30	**49.50**	**48.70**	−0.80
	InternLM2.5-7B	44.30	48.00	**52.50**	**54.10**	+1.60

3.6 A5: Given the exhaustive table, does the objective matter?

With the table in hand, the objective still matters. Table 4 trains the same selector with three supervised alternatives. Cross-entropy on the reward-optimal subset sees one label per question and collapses toward the empty set, as SFT does in Table 2; a softmax target over the sixteen rewards helps but still calls under half a tool per question (FGPO is significantly better in 15 and 14 of the 15 cells respectively). Regressing all sixteen rewards with a value head is the strongest alternative, yet FGPO is significantly better in all ten GenomeQA and GenBench-X cells and tied in all five BM4 cells. Full action-level feedback helps, but the objective used to exploit it also matters.

4 Related Work

Tool-augmented genomic reasoning. Genomic foundation models learn representations of nucleotide sequences (Ji et al., 2021; Zhou et al., 2024; Dalla-Torre et al., 2025; Nguyen et al., 2023; Schiff et al., 2024; Nguyen et al., 2024), and general-purpose LLMs benefit from such tools (Jin et al., 2024); public genomic benchmarks exist (Zhou et al., 2024; Grešová et al., 2023; De Almeida et al., 2022). Our focus is the tool interface: which genomic computations to expose per question.

RL-based tool selection. Teaching LLMs to call external tools spans self-supervised call insertion (Schick et al., 2023), prompted acting (Yao et al., 2022; Lu et al., 2023), API-scale instruction tuning (Qin et al., 2024; Patil et al., 2024; Shen et al., 2023), and benchmarks of call correctness (Li et al., 2023). Closest to us are methods that train a *selection* policy for a frozen reasoner with RL: VisTA (Huang et al., 2025), AuTAgent (Tong et al., 2026), and reward-shaped variants (Qian et al., 2026; Jin et al., 2025), all modifying the reward or interaction protocol but keeping sampled policy optimization. We adopt their frozen-reasoner setting but study the combinatorial subset-selection regime: what happens when the full tool-subset space is small enough to evaluate but the optimizer still samples it? Integrating over actions rather than sampling them has been explored for variance reduction (Ciosek & Whiteson, 2018; 2020; Asadi et al., 2017; Kool et al., 2019; Williams, 1992; Ahmadian et al., 2024). GRPO (Shao et al., 2024) and its successors (Yu et al., 2026; Liu et al., 2025) normalize within sampled groups; the resulting zero-advantage groups are noted as an efficiency concern (Yu et al., 2026). We characterize its interaction with policy concentration on enumerable spaces, where it becomes a structural mismatch, not a mere inefficiency, and propose FGPO for this regime.

5 Conclusion and Future Work

On small enumerable spaces, sampled group optimization starves as the policy concentrates: rewards collide and advantages vanish. FGPO optimizes the exact action expectation, improving accuracy on three genomic benchmarks and five frozen reasoners with fewer tools. While compact tool libraries suffice in the specialist regime studied here, future work could extend FGPO to larger libraries by adaptively constructing coverage-preserving candidate sets and optimizing exactly within each.

AI use statement

We have not used generative AI tools for any of the tasks with required disclosure, and the remaining required disclosure tasks are not applicable to this work. Additionally, we used generative AI tools for polishing the writing and improving the readability of the manuscript. We have reviewed all AI-assisted work: all AI-assisted edits were checked against the underlying results by the authors. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

Ethics statement

This work involves no human subjects and no clinical or individually identifiable genomic data. All sequences come from public reference-organism corpora; the two held-out evaluation suites are derivatives of those corpora and redistribution should respect each upstream licence. The method learns only which standard sequence-analysis tools to invoke and neither designs sequences nor adds capabilities beyond those the tools already provide.

Reproducibility statement

Anonymized code for FGPO (reward-table construction, training, and both the free-generation and candidate-scoring evaluations) is included in the supplementary material. The remaining training and evaluation code (baselines and analyses), the exhaustive reward tables, per-question answer files for every table row, and the failure log (including models that could not be evaluated on our hardware, with root causes) will be released. Baselines use official TRL implementations (von Werra et al., 2020); every number in the paper can be regenerated by scripted lookup from the tables to be released.

References

Arash Ahmadian, Chris Cremer, Matthias Gallé, Marzieh Fadaee, Julia Kreutzer, Olivier Pietquin, Ahmet Üstün, and Sara Hooker. Back to basics: Revisiting reinforce-style optimization for learning from human feedback in llms. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 12248–12267, 2024.

Kavosh Asadi, Cameron Allen, Melrose Roderick, Abdel-rahman Mohamed, George Konidaris, Michael Littman, and Brown University Amazon. Mean actor critic. *stat*, 1050(2017):1, 2017.

Zheng Cai, Maosong Cao, Haojiong Chen, Kai Chen, Keyu Chen, Xin Chen, Xun Chen, Zehui Chen, Zhi Chen, Pei Chu, et al. Internlm2 technical report. *arXiv preprint arXiv:2403.17297*, 2024.

Kamil Ciosek and Shimon Whiteson. Expected policy gradients, 2018. URL https://arxiv.org/abs/1706.05374.

Kamil Ciosek and Shimon Whiteson. Expected policy gradients for reinforcement learning. *Journal of Machine Learning Research*, 21(52):1–51, 2020.

Peter JA Cock, Tiago Antao, Jeffrey T Chang, Brad A Chapman, Cymon J Cox, Andrew Dalke, Iddo Friedberg, Thomas Hamelryck, Frank Kauff, Bartek Wilczynski, et al. Biopython: freely available python tools for computational molecular biology and bioinformatics. *Bioinformatics*, 25(11):1422, 2009.

Hugo Dalla-Torre, Liam Gonzalez, Javier Mendoza-Revilla, Nicolas Lopez Carranza, Adam Henryk Grzywaczewski, Francesco Oteri, Christian Dallago, Evan Trop, Bernardo P De Almeida, Hassan Sirelkhatim, et al. Nucleotide transformer: building and evaluating robust foundation models for human genomics. *Nature methods*, 22(2):287–297, 2025.

Bernardo P De Almeida, Franziska Reiter, Michaela Pagani, and Alexander Stark. Deepstarr predicts enhancer activity from dna sequence and enables the de novo design of synthetic enhancers. *Nature genetics*, 54(5):613–624, 2022.

Thomas G Dietterich. Approximate statistical tests for comparing supervised classification learning algorithms. *Neural computation*, 10(7):1895–1923, 1998.

Katarína Grešová, Vlastimil Martinek, David Čechák, Petr Šimeček, and Panagiotis Alexiou. Genomic benchmarks: a collection of datasets for genomic sequence classification. *BMC Genomic Data*, 24 (1):25, 2023.

Charles R Harris, K Jarrod Millman, Stéfan J Van Der Walt, Ralf Gommers, Pauli Virtanen, David Cournapeau, Eric Wieser, Julian Taylor, Sebastian Berg, Nathaniel J Smith, et al. Array programming with numpy. *nature*, 585(7825):357–362, 2020.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*, 2021.

Zeyi Huang, Yuyang Ji, Anirudh Sundara Rajan, Zefan Cai, Wen Xiao, Haohan Wang, Junjie Hu, and Yong Jae Lee. Visualtoolagent (vista): A reinforcement learning framework for visual tool selection. *arXiv preprint arXiv:2505.20289*, 2025.

Yanrong Ji, Zhihan Zhou, Han Liu, and Ramana V Davuluri. Dnabert: pre-trained bidirectional encoder representations from transformers model for dna-language in genome. *Bioinformatics*, 37 (15):2112–2120, 2021.

Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, Lélio Renard Lavaud, Marie-Anne Lachaux, Pierre Stock, Teven Le Scao, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. Mistral 7b, 2023. URL https://arxiv.org/abs/2310.06825.

Bowen Jin, Hansi Zeng, Zhenrui Yue, Jinsung Yoon, Sercan Arik, Dong Wang, Hamed Zamani, and Jiawei Han. Search-r1: Training llms to reason and leverage search engines with reinforcement learning. *arXiv preprint arXiv:2503.09516*, 2025.

Qiao Jin, Yifan Yang, Qingyu Chen, and Zhiyong Lu. Genegpt: augmenting large language models with domain tools for improved access to biomedical information. *Bioinformatics*, 40(2):btae075, 2024.

Wouter Kool, Herke Van Hoof, and Max Welling. Buy 4 reinforce samples, get a baseline for free! 2019.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph Gonzalez, Hao Zhang, and Ion Stoica. Efficient memory management for large language model serving with pagedattention. In *Proceedings of the 29th symposium on operating systems principles*, pp. 611–626, 2023.

John Langford and Tong Zhang. The epoch-greedy algorithm for multi-armed bandits with side information. *Advances in neural information processing systems*, 20, 2007.

Minghao Li, Yingxiu Zhao, Bowen Yu, Feifan Song, Hangyu Li, Haiyang Yu, Zhoujun Li, Fei Huang, and Yongbin Li. Api-bank: A comprehensive benchmark for tool-augmented llms. In *Proceedings of the 2023 conference on empirical methods in natural language processing*, pp. 3102–3116, 2023.

Zichen Liu, Changyu Chen, Wenjun Li, Penghui Qi, Tianyu Pang, Chao Du, Wee Sun Lee, and Min Lin. Understanding r1-zero-like training: A critical perspective. *arXiv preprint arXiv:2503.20783*, 2025.

Weicai Long, Yusen Hou, Junning Feng, Shuo Yang, Donglin Xie, Yanlin Zhang, et al. Genomeqa: Benchmarking general large language models for genome sequence understanding. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 35771–35792, 2026.

Pan Lu, Baolin Peng, Hao Cheng, Michel Galley, Kai-Wei Chang, Ying Nian Wu, Song-Chun Zhu, and Jianfeng Gao. Chameleon: Plug-and-play compositional reasoning with large language models. *Advances in Neural Information Processing Systems*, 36:43447–43478, 2023.

Albert W Marshall, Ingram Olkin, and Barry C Arnold. *Inequalities: theory of majorization and its applications*. Springer, 1979.

Quinn McNemar. Note on the sampling error of the difference between correlated proportions or percentages. *Psychometrika*, 12(2):153–157, 1947.

Eric Nguyen, Michael Poli, Marjan Faizi, Armin Thomas, Michael Wornow, Callum Birch-Sykes, Stefano Massaroli, Aman Patel, Clayton Rabideau, Yoshua Bengio, et al. Hyenadna: Long-range genomic sequence modeling at single nucleotide resolution. *Advances in neural information processing systems*, 36:43177–43201, 2023.

Eric Nguyen, Michael Poli, Matthew G Durrant, Brian Kang, Dhruva Katrekar, David B Li, Liam J Bartie, Armin W Thomas, Samuel H King, Garyk Brixi, et al. Sequence modeling and design from molecular to genome scale with evo. *Science*, 386(6723):eado9336, 2024.

Shishir G Patil, Tianjun Zhang, Xin Wang, and Joseph E Gonzalez. Gorilla: Large language model connected with massive apis. *Advances in Neural Information Processing Systems*, 37: 126544–126565, 2024.

Cheng Qian, Emre Can Acikgoz, Qi He, Hongru Wang, Xiusi Chen, Dilek Hakkani-Tur, Gokhan Tur, and Heng Ji. Toolrl: Reward is all tool learning needs. *Advances in Neural Information Processing Systems*, 38:105523–105553, 2026.

Yujia Qin, Shihao Liang, Yining Ye, Kunlun Zhu, Lan Yan, Yaxi Lu, Yankai Lin, Xin Cong, Xiangru Tang, Bill Qian, et al. Toolllm: Facilitating large language models to master 16000+ real-world apis. In *International Conference on Learning Representations*, volume 2024, pp. 9695–9717, 2024.

Qwen, :, An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jingren Zhou, Junyang Lin, Kai Dang, Keming Lu, Keqin Bao, Kexin Yang, Le Yu, Mei Li, Mingfeng Xue, Pei Zhang, Qin Zhu, Rui Men, Runji Lin, Tianhao Li, Tianyi Tang, Tingyu Xia, Xingzhang Ren, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yu Wan, Yuqiong Liu, Zeyu Cui, Zhenru Zhang, and Zihan Qiu. Qwen2.5 technical report, 2025. URL https://arxiv.org/abs/2412.15115.

Ieva Rauluseviciute, Rafael Riudavets-Puig, Romain Blanc-Mathieu, Jaime A Castro-Mondragon, Katalin Ferenc, Vipin Kumar, Roza Berhanu Lemma, Jérémy Lucas, Jeanne Chèneby, Damir Baranasic, et al. Jaspar 2024: 20th anniversary of the open-access database of transcription factor binding profiles. *Nucleic acids research*, 52(D1):D174–D182, 2024.

Vyzantinos Repantis, Ameya Gawde, Harshvardhan Singh, and Joey Blackwell II. How many tools should an llm agent see? a chance-corrected answer. *arXiv preprint arXiv:2605.24660*, 2026.

Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Eric Hambro, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. Toolformer: Language models can teach themselves to use tools. *Advances in neural information processing systems*, 36:68539–68551, 2023.

Yair Schiff, Chia-Hsiang Kao, Aaron Gokaslan, Tri Dao, Albert Gu, and Volodymyr Kuleshov. Caduceus: Bi-directional equivariant long-range dna sequence modeling. *Proceedings of machine learning research*, 235:43632, 2024.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, YK Li, Yang Wu, et al. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024.

Yongliang Shen, Kaitao Song, Xu Tan, Dongsheng Li, Weiming Lu, and Yueting Zhuang. Hugginggpt: Solving ai tasks with chatgpt and its friends in hugging face. *Advances in Neural Information Processing Systems*, 36:38154–38180, 2023.

Siqian Tong, Xuan Li, Yiwei Wang, Baolong Bi, Yujun Cai, Shenghua Liu, Yuchen He, and Chengpeng Hao. Autagent: A reinforcement learning framework for tool-augmented audio reasoning. *arXiv preprint arXiv:2602.13685*, 2026.

Leandro von Werra, Younes Belkada, Lewis Tunstall, Edward Beeching, Tristan Thrush, Nathan Lambert, Shengyi Huang, Kashif Rasul, and Quentin Gallouédec. TRL: Transformers Reinforcement Learning, 2020. URL https://github.com/huggingface/trl.

Ronald J Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3):229–256, 1992.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models. *arXiv preprint arXiv:2210.03629*, 2022.

Gene Yeo and Christopher B Burge. Maximum entropy modeling of short sequence motifs with applications to rna splicing signals. In *Proceedings of the seventh annual international conference on Research in computational molecular biology*, pp. 322–331, 2003.

Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, Weinan Dai, Tiantian Fan, Gaohong Liu, Lingjun Liu, et al. Dapo: An open-source llm reinforcement learning system at scale. *Advances in Neural Information Processing Systems*, 38:113222–113244, 2026.

Zhihan Zhou, Yanrong Ji, Weijian Li, Pratik Dutta, Ramana Davuluri, and Han Liu. Dnabert-2: Efficient foundation model and benchmark for multi-species genomes. In *International Conference on Learning Representations*, volume 2024, pp. 41642–41665, 2024.

Contents of Appendix

A Algorithm **15**

B Experiment Settings **15**

B.1 Prompts and action encoding 15

B.2 Sensitivity checks 16

B.3 A five-tool library: the policy rejects a harmful capability 17

B.4 Hyperparameters 17

B.5 Anonymous slot encoding examples 17

B.6 Baselines 18

B.7 Reward 18

B.8 Benchmark construction and split hygiene 19

B.9 Tool implementations 19

B.10 Infrastructure and models that could not be evaluated 19

C Full Transfer Results **19**

C.1 Case study 19

C.2 Where the gain comes from 19

C.3 The transfer grid, read as shapes 20

C.4 Pipeline agreement with the exhaustive table 22

C.5 Ceilings: the best fixed subset and the per-question oracle 22

C.6 Paired significance 23

D Additional Analyses **23**

D.1 k-subset interpolation in full 23

D.2 Per-question paired analysis and selection heatmap 23

D.3 Data-scale ablation 23

D.4 Effect of parsimony on accuracy and tool usage 24

D.5 Tool-library single-removal ablations 24

E Baseline Fairness Sweeps **25**

E.1 GRPO checkpoint and configuration sweeps 25

E.2 DPO β sweep 25

E.3 GRPO and DAPO trained on Eq. 1 26

F Mechanism Details **26**

F.1 Two properties of dead groups and of the exact objective 26

F.2 Dead-group grids for all three benchmarks 27

F.3 Empty-subset preference: how offline objectives collapse 27

G Prompt Templates **28**

G.1 Policy prompt (all trained arms: FGPO, GRPO, SFT, DPO) 28

G.2 Prompted baseline (Tools w/ Desc) 29

G.3 Reasoner prompt 29

G.4 Complete reasoner prompt 29

G.5 Complete tool evidence examples 29

G.6 Question format examples 30

G.7 Tool description prompt (Tools w/ Desc baseline) 31

H Additional Case Studies **32**

H.1 One rescue per benchmark and task family 32

H.2 Walk-through: GenomeQA, TF motif 34

H.3 Walk-through: GenBench-X, `human_ensembl_regulatory` 36

H.4 Walk-through: BM4, bacterial σ^{70} promoters 37

I Further Examples **38**

I.1 Where each policy puts its probability 38

I.2 Fewer tools, right answer 38

I.3 Failures of FGPO 42

I.4 Walk-through: fewer tools (GenomeQA, promoter/enhancer) 42

A Algorithm

Algorithm 1 states FGPO end to end. Stage 1 is the one-off cost (Section 2.4); Stage 2 touches the frozen reasoner not at all, since every reward it needs is a table lookup. The inner loop scores all 2^n candidate strings in a single batched forward pass, requiring no policy-side sampling.

B Experiment Settings

B.1 Prompts and action encoding

The policy prompt lists n anonymous tool slots ("`0: type1 (A)`" style) and the question; the answer is an indexed subset in `<answer></answer>` tags. Tool names, descriptions, and any "useful-for" hints are withheld from the policy in every trained arm (FGPO, GRPO, SFT, DPO); the Tools-w/-Desc baseline receives real names and functional descriptions. Reasoner prompts concatenate the question, options, and the rendered evidence of the selected subset; evidence rendering is shared verbatim across every method and every table row. **Anonymous slots are not a handicap.** To check that the policy learns routing from reward rather than from tool names, we retrain FGPO with the real tool names and functional descriptions in its selection prompt. On a fixed 1,200-question subsample of the GenomeQA test set the two are nearly identical: 51.67 with descriptions against 51.75 without. Format rates (the share of completions whose answer tag contains only in-range slot indices; the same parser scores every free-generation arm) are 1.00 and 0.79 respectively; the gap is the out-of-range index discussed next. The anonymous encoding used throughout therefore costs nothing, and the routing reported in Section 3.3 cannot have been read off the prompt.

Out-of-range slot indices. Free generation can emit an index outside the four slots. The reported FGPO checkpoint does so on 747 of the 3,590 GenomeQA questions, 45 of the 1,000 BM4 questions and 1 of the 1,000 GenBench-X questions, almost always as `<answer>3,4</answer>`

Algorithm 1: FGPO (Full-Group Policy Optimization)

Input: questions $\mathcal{D}$; tool library $\{T_1, \ldots, T_n\}$; frozen reasoner R; LoRA-parameterized policy q_θ; entropy weight β; parsimony weight λ
Output: trained selector q_θ

```
// Stage 1:  exhaustive reward table (once)
```
foreach $s \in \mathcal{D}$ **do**
  foreach $a \in \mathcal{A} = 2^{\{T_1,\ldots,T_n\}}$ **do**
    run the tools in a on s's sequence; query R with the rendered evidence
    store $r(s, a)$ by Eq. 1
  end
end
```
// Stage 2:  exact policy optimization
```
for *epoch* $= 1, \ldots, E$ **do**
  foreach *minibatch* $B \subseteq \mathcal{D}$ **do**
    foreach $s \in B$ **do**
      score every candidate string: $\ell_\theta(a \mid s)$ for all $a \in \mathcal{A}$ (Eq. 4) `// one batched forward pass`
      $q_\theta(\cdot \mid s) \leftarrow \mathrm{softmax}_{a \in \mathcal{A}}\, \ell_\theta(a \mid s)$
    end
    $\mathcal{L}(\theta) \leftarrow -\frac{1}{|B|} \sum_{s \in B} \left[\sum_{a \in \mathcal{A}} q_\theta(a|s)\, r(s, a) + \beta\, H\big(q_\theta(\cdot|s)\big) \right]$ `// Eq. 2, 5`
    $\theta \leftarrow \theta - \eta\, \nabla_\theta \mathcal{L}(\theta)$ `// every a contributes; no group, no dead groups`
  end
  select the checkpoint on the held-out development split
end

or <answer>2,3,4</answer> (one GenomeQA completion is <answer>4</answer>); the parser drops the nonexistent index 4 and keeps the valid ones, so these are scored as $\{T_4\}$, $\{T_3, T_4\}$ and the empty subset respectively, and its GenomeQA format rate is 0.79. GRPO's completions never do this. The candidate-scoring rule of Section 2.3 cannot produce such outputs, and under it FGPO still leads GRPO by 6.71 points.

B.2 Sensitivity checks

Table 5 varies three choices one at a time on a fixed 1,200-question subsample of the GenomeQA test set, disjoint from the 492-question development split used for checkpoint selection. The exact objective changes little under the tested entropy coefficient, since raising β from 0.03 to 0.08 changes the diagnostic peak by only 0.25 points, and under a sharper listwise softmax, $q_\tau(a \mid s) \propto \exp\big(\ell_\theta(a \mid s)/\tau\big)$ with τ=0.7 in place of Eq. 4. A 1.5B selector in place of the 7B one still reaches 50.42 against six-rollout GRPO's 44.25, so the advantage is not specific to the 7B selector.

Table 5: **Sensitivity of the exact objective.** Accuracy (%) per epoch on the 1,200-question subsample of the GenomeQA test set; *peak* is the maximum observed accuracy on this diagnostic subsample and is reported only for sensitivity analysis, never for checkpoint selection. Every row changes one factor from the reported configuration (first row). GRPO scores 44.25 here; a dash marks a cell not evaluated on this subsample.

Variant	ep1	ep2	ep3	ep4	peak
Reported (β=0.03, 7B policy)	47.00	48.83	**51.75**	—	**51.75**
Entropy $\beta = 0.08$	48.17	50.50	**52.00**	50.67	52.00
Listwise softmax temperature 0.7	46.00	48.75	**50.92**	50.58	50.92
1.5B policy (same objective)	47.33	**50.42**	48.42	49.58	50.42

B.3 A FIVE-TOOL LIBRARY: THE POLICY REJECTS A HARMFUL CAPABILITY

To probe what happens when the library grows, we add a fifth tool T_5, a homology-search module, while keeping T_1–T_4 in place, and retrain the exact objective over the resulting $2^5 = 32$ subsets. T_5 searches the query against a local database of labelled sequences by k-mer seeding followed by Smith–Waterman alignment (Cock et al., 2009), reporting the closest matches with label, percent identity and query coverage; the database is built from training and development sequences only, and self-hits and near-identical matches ($> 99.5\%$ identity) are dropped so the tool cannot retrieve the query and hand back its own label. This tool is a poor fit for the benchmark: used alone it scores 34.07, below the 39.33 no-tool floor, so a selector that simply invokes everything available should be harmed by it. On the full $3{,}590$-question test set the five-tool policy reaches 45.38 at its final checkpoint, above exhaustive invocation of all five tools (44.51) and above the best post-hoc fixed subset (45.18); accuracy rises monotonically over the four epochs ($43.04 \to 44.15 \to 44.51 \to 45.38$), so the final checkpoint is also the best observed one; no checkpoint is selected using test performance. The per-tool invocation rates explain why: the policy calls the motif scanner on 61.9% of questions and the genomic encoder on 14.3%, but the harmful new module on only 2.8%, and composition statistics, the weakest of the original four, on 0.0%. Learning from complete action-level feedback therefore includes learning which capabilities to *refuse*, not only which to prefer. These runs use a separately constructed reward table for the five-tool action space, and every reference point quoted above (34.07, 39.33, 44.51, 45.18) is read from that same 32-mask table. That table was built with an earlier version of the genomic encoder T_4: on the same $3{,}590$ questions, the subsets without T_4 agree with the four-tool table to within 0.1 points, while every subset containing it scores lower (T_4 alone 40.25 vs. 47.77). All comparisons are therefore internal to this table, and the runs are intended as a library-expansion stress test rather than a matched performance comparison with the four-tool setting of Table 2.

B.4 HYPERPARAMETERS

Table 6 lists every value; Appendix B.2 reports what happens when they are varied. Three details do not fit the table. The reported GRPO run is the configuration that faithfully reproduces Tong et al.'s published setting, fixed *a priori* rather than selected by score; four further independent configurations (varying G, temperature and data mix) span 43.90–45.57. And checkpoint selection for FGPO and SFT uses the same 492-question development split, with test accuracy reported at the dev-selected checkpoint; for FGPO, dev and test agree on the peak (ep3: 52.24 dev / 52.62 test). GRPO is reported at its best checkpoint (step 75). For DPO, every β and every checkpoint collapses to the no-tool floor, so we report the best cell of its entire sweep, which is more generous than dev selection and still leaves it at 39.33. The five-tool run of Appendix B.3 has no development sweep of its own and is reported at its final epoch. FGPO, SFT and DPO train for 4 epochs and GRPO for 200 steps (≈ 6.3 epochs), each under its own schedule rather than a matched epoch count. GRPO's accuracy is flat from step 50 onward (44.85–45.46, Appendix E.1). To check that the shorter FGPO schedule does not decide the comparison, we continued FGPO from its epoch-4 checkpoint for two more epochs (fresh optimizer state, new data order): it scores 51.34 and 52.67 on GenomeQA (epochs 5–6 in Figure 7d), level with the reported 52.62. These two checkpoints enter no table and no checkpoint selection.

The trained arms share the same backbone, LoRA configuration, training data, prompt format, and data order; objective-specific optimization settings and training durations are listed explicitly below.

B.5 ANONYMOUS SLOT ENCODING EXAMPLES

To verify that the policy cannot read tool identity from the prompt, we show the full input-output pair for one training question.

Policy input.

```
[System] You are an expert agent specialized in selecting tools to
solve genomic reasoning tasks.  You are provided with access to 4
tools, indexed from 0 to 3.

[User] Available tools:
0:  type1 (A)
```

Table 6: Key hyperparameters. "GRPO arms" are GRPO, GRPO‡ and DAPO‡ (Appendix E.3).

Hyperparameter	Value	Shared across arms?
Base model	Qwen2.5-7B-Instruct	✓
LoRA rank / alpha	16 / 32	✓
Learning rate	10^{-5}	✓
FGPO/SFT/DPO batch size	16 prompts	FGPO/SFT/DPO
GRPO prompts per step	64 ($\times 6$ rollouts)	GRPO arms
FGPO/SFT/DPO training	4 epochs	FGPO/SFT/DPO
GRPO training	200 steps ($\approx$ 6.26 epochs as logged by the trainer)	GRPO arms
Gradient checkpointing	Yes	✓
Entropy bonus β (FGPO)	0.03	—
Parsimony λ (Eq. 1)	0.10	FGPO/SFT/DPO/GRPO‡/DAPO‡
GRPO rollouts G	6	GRPO arms
GRPO temperature	1.0	GRPO arms
DPO β	0.3 (all of 0.1/0.3/0.5 collapse)	DPO only
Controlled candidate-scoring eval.	per-token mean (Eq. 4)	all trained arms
Dev-set checkpoint selection	492 questions	FGPO/SFT/GRPO‡/DAPO‡

```
1:  type2 (B)
2:  type3 (C)
3:  type4 (D)

I have extracted a Human DNA sequence...Which of the following best
describes it?
A: enhancer region   B: promoter region   C: splice site   D: coding
region

Select the index number(s)...enclosed in <answer></answer> tags.
```

Policy output (FGPO, converged).

```
<answer>3</answer>
```

Tool 3 maps to T_4 (genomic expert). The policy has no way to know this from the prompt: it learned the mapping entirely from reward.

Policy output (GRPO, converged).

```
<answer>0,1,2,3</answer>
```

The converged GRPO policy selects all four tools on 19% of GenomeQA questions (Figure 4).

B.6 Baselines

Random draws one subset uniformly from all 2^n, including the empty one, per question at a fixed seed. *All-Tools* always invokes the full library. *Tools w/ Desc* is the same unadapted Qwen2.5-7B-Instruct backbone used as the policy, given the real tool names and functional descriptions in its selection prompt. GRPO is evaluated at checkpoints $\{25, \ldots, 200\}$ and reported at its best checkpoint (step 75); DPO at the best of a $\beta \in \{0.1, 0.3, 0.5\}$ sweep over seven checkpoints, taking the best cell outright rather than a dev-selected one (Appendix E).

B.7 Reward

Eq. 1: ± 1 on the correctness of the reasoner's answer under the subset, plus $\lambda(1 - 2|a|/n)$ with $\lambda = 0.10$, so the reward runs in $[-1.1, +1.1]$ and parsimony can never outrank being right. The GRPO arm instead uses the differential reward of Tong et al. (2026),

$$r_{\text{diff}}(s, a) = \mathbf{1}[\hat{y}(s, a) = y^\star] - \mathbf{1}[\hat{y}(s, \emptyset) = y^\star] \in \{-1, 0, +1\},$$

i.e. $+1$ when tools rescue a wrong no-tool answer, -1 when they break a right one and 0 otherwise, combined as $0.95\, r_{\text{diff}} + 0.05\, r_{\text{fmt}}$ with $r_{\text{fmt}} \in \{0, 1\}$ marking a parseable completion, faithful to that

paper. Every one of the 2^n enumerated candidates is parseable by construction, so r_{fmt} is constant over them and does not change the reward partition used in the closed-form dead-group analysis. The binary arm of Table 12 is Eq. 1 with $\lambda = 0$: ± 1 on correctness, no parsimony term. Every arm reuses the same precomputed frozen-reasoner outcomes for all subsets of every training question (Section 2.4); FGPO, SFT and DPO derive Eq. 1 from them, GRPO derives its differential reward.

B.8 Benchmark construction and split hygiene

For *GenomeQA*, evaluation uses the promoter/enhancer, splice-site, taxonomy, histone-mark and TF-motif task families. *Training questions* are built separately from the Nucleotide Transformer downstream tasks (enhancers, promoter_all, splice_sites_all, four histone marks) in the same question shapes GenomeQA uses, so taxonomy and TF-motif families are first seen at test time. *GenBench-X* draws 200 questions from each of five Genomic Benchmarks subsets: `demo_human_or_worm`, `human_ensembl_regulatory`, `human_ocr_ensembl`, `drosophila_enhancers_stark`, `demo_coding_vs_intergenomic`, two of them non-human. *BM4* draws 250 from each of four sources: bacterial σ^{70} promoters, a bacteria-vs-archaea taxonomy set, DeepSTARR fly enhancers, and mouse Ensembl enhancers. Each GenBench-X and BM4 source is filed under one of GenomeQA's five task slots, the label under which T_4 keeps that source's reference set (histone mark: `human_ocr_ensembl`, mouse enhancers; promoter/enhancer: `human_ensembl_regulatory`, σ^{70} promoters; splice site: `demo_coding_vs_intergenomic`; taxonomy: `demo_human_or_worm`, bacteria vs. archaea; TF motif: `drosophila_enhancers_stark`, DeepSTARR). Integer labels carry no semantics in these corpora, so every label mapping was established by measurement (GC content, in-frame stop-codon depletion, CpG observed/expected, cross-subset sequence identity) rather than from the dataset name; two turn out to be the reverse of what the name suggests. Sequences beyond 1,000 bp are centre-cropped to GenomeQA's ceiling. Sequence-level disjointness between the training pool and each evaluation suite is checked before any file is written, and each task slot's kNN reference set is rebuilt from that source's own training split.

B.9 Tool implementations

T_1 *sequence composition*: GC content and k-mer statistics via NumPy; nothing trained. T_2 *motif scanning*: JASPAR 2024 matrices (Rauluseviciute et al., 2024) from `pyjaspar`, scanned with a Biopython PSSM scan (Cock et al., 2009). T_3 *splice-site analysis*: GT/AG only enumerates candidate positions; MaxEntScan scores them (9-mer donor, 23-mer acceptor models). T_4 *genomic expert*: mean-pooled embeddings from a frozen `nucleotide-transformer-v2-50m-multi-species` encoder (Dalla-Torre et al., 2025) with a cosine-metric kNN (k=15) over a labelled reference set built from the corresponding task's *training* split only, with neither test sequences nor test labels included in the reference set, and the encoder is never fine-tuned. Every tool renders into the same structured evidence block for every method and table row.

B.10 Infrastructure and models that could not be evaluated

All experiments ran on Ascend 910B NPUs with vLLM-Ascend serving (Kwon et al., 2023). For reproducibility we record models excluded for stack reasons: GLM-4-9B (fused-RoPE kernel lacks partial-rotary support), Gemma-2-9B (`rope_theta` config incompatibility in the serving stack), Yi-1.5-9B (weights unavailable). None were excluded for score reasons.

C Full Transfer Results

C.1 Case study

Table 7 shows three representative questions where the frozen reasoner fails without tools but is rescued by FGPO's tool selection.

C.2 Where the gain comes from

Table 9 splits the GenomeQA column of Table 2 into its five task families. Two things are visible only at this resolution. First, the gain is not uniform: FGPO adds 20.8 points over the unaided reasoner on

Table 7: **Case study.** Three questions the reasoner gets wrong, rescued by FGPO's tool call. Questions and tool outputs are quoted from the exhaustive reward table; interpretations are summarized for clarity.

Qwen3-8B (No-Tool)	FGPO (Ours)	
Wrong Answer	**Tool Call**	**Final Answer**
Ex1 (TF motif): "Which Human DNA sequence is a target for SRF?" [GT: D]		
Picks A; nothing separates the candidates.	`motif_scanner:` {A: JUN, FOS, TEAD4 | B: STAT1 | C: SP1 | **D: SRF**}	D carries the named motif.
Ex2 (Taxonomy): "Select the DNA sequence derived from a Virus genome." [GT: B]		
Picks A, the most eukaryotic.	`genomic_expert:` {A: Euk .82 | **B: Virus .37** | C: Euk .53 | D: Euk .84}	B is the only non-eukaryote.
Ex3 (Splice site): "Which Human DNA sequence contains a functional Only Acceptor?" [GT: B]		
Picks A, which has no splice site at all.	`genomic_expert:` {A: none | **B: Only Acceptor .53** | C: none | D: none}	B is the only candidate with a site.

Table 8: **Accuracy and cost.** Upper: all methods on GenomeQA under the training reasoner. Lower: FGPO against exhaustive invocation on all three benchmarks. Input tokens count question + rendered evidence, measured for FGPO and All-Tools.

GenomeQA	No-Tool	GRPO	Tools w/ Desc	All-Tools	FGPO (ours)
Accuracy	39.28	45.46	47.74	49.39	**52.62**
Tools / question	0.00	2.36	1.03	4.00	1.40
Input tokens	–	–	–	4,486	1,300

FGPO vs. All-Tools	GenomeQA	GenBench-X	BM4
FGPO tool calls	1.40	1.58	1.70
Call ratio	2.86×	2.54×	2.36×
Input-token ratio	3.45×	4.14×	3.65×

taxonomy and 22.7 on TF motif, but only 3.3 on splice sites and 3.8 on histone marks, the families where the per-question oracle itself is lowest (68.7 and 67.7), so the library simply carries less signal there. Second, the average number of tools the policy calls *varies* by family, from 1.18 on taxonomy to 1.51 on promoter/enhancer and 1.53 on splice sites, so the policy does not spend a uniform budget. Nothing in the reward asks for this (Eq. 1 penalises tools uniformly), so it is a property the exact objective discovers rather than one it is told.

Table 9: **GenomeQA by task family** (n=718 each, Qwen3-8B reasoner). Accuracies (%); *tools* is FGPO's mean subset size on that family.

Family	No-Tool	Tools w/ Desc	GRPO	FGPO	Oracle
Taxonomy	46.10	55.15	56.55	**66.85**	81.6
TF motif	38.86	61.42	46.80	**61.56**	85.8
Promoter/enhancer	35.93	35.65	43.31	**52.09**	83.3
Splice site	38.16	**44.71**	40.53	41.50	68.7
Histone mark	37.33	**41.78**	40.11	41.09	67.7
FGPO mean tools: taxonomy 1.18, TF motif 1.32, prom./enh. 1.51, histone 1.45, splice 1.53					

C.3 The transfer grid, read as shapes

The four transfer columns of Table 2 were produced by transplanting the policy trained against Qwen3-8B rewards into a different frozen reasoner, with no adaptation, no retuning and no access to the new reasoner during training. That this works at all is not obvious, since a selection policy could easily have learned which subsets suit one particular reader, so the result worth stating is the

margin's consistency rather than its size: FGPO−GRPO is positive in all 15 reasoner×benchmark cells (min $+0.92$, median $+4.48$, max $+14.20$; Figure 9), and no reasoner and no benchmark supplies a counterexample. DPO tracks the no-tool floor within 0.41 points in all 15 cells, consistent with its empirical collapse onto the empty subset, which turns that collapse into a five-reasoner-wide measurement rather than a single-column anecdote.

Why not curate a fixed library once and skip the policy? Because the answer is unstable and cannot price optionality: the best fixed subset differs by benchmark (T_4 on GenBench-X, $T_3{+}T_4$ on BM4, $T_2{+}T_4$ on GenomeQA), and the tools a fixed subset omits can still be the only ones that answer particular questions correctly (Appendix C.5).

The numbers themselves are in Table 2; this section reads their shape. Figure 8 plots the same grid per benchmark, which makes two things visible that a table of 135 numbers does not. First, more evidence is not monotonically better: on Mistral-7B's GenomeQA column, invoking the whole library scores 23.09 against a no-tool floor of 37.83, i.e. 14.7 points of damage done purely by handing the reasoner more to read. Second, FGPO tracks BestFixed*, a test-label-selected fixed reference that needs test labels to pick its subset, closely enough that the gap is two-signed, which is what one expects when a single fixed subset happens to be near-optimal for a given reasoner.

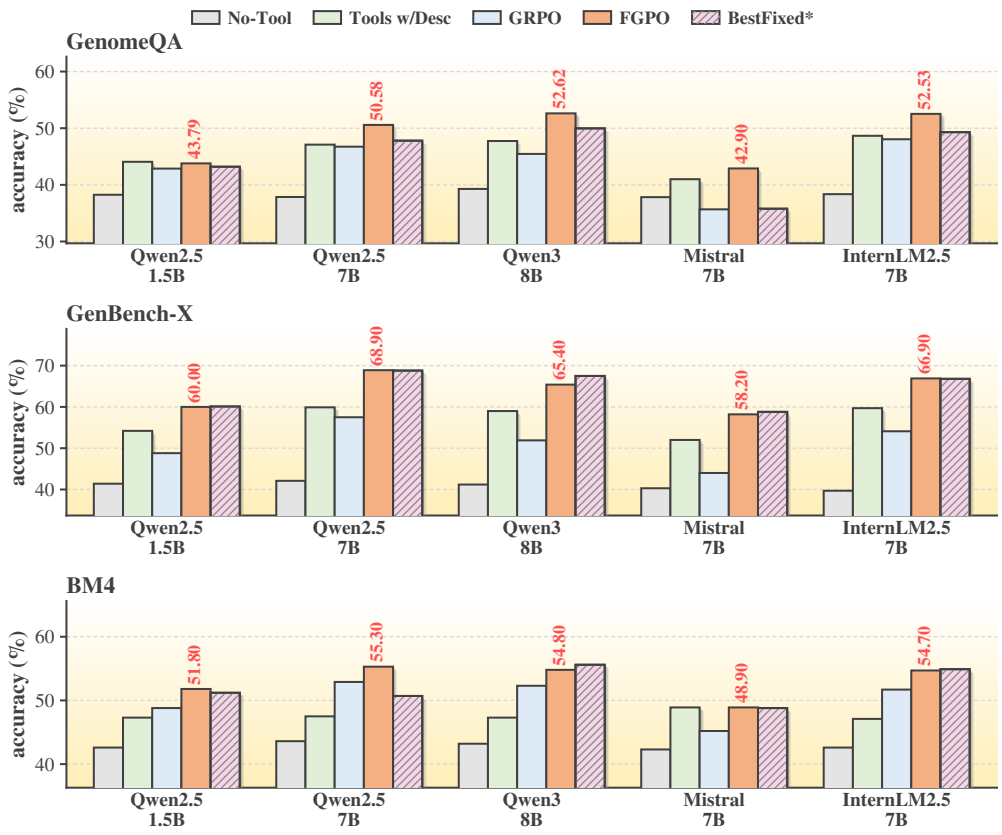


Figure 8: Plug-in transfer per benchmark across five frozen reasoners (the grid of Table 2). Random, SFT and DPO are omitted here for legibility and reported in full in the table.

Figure 9 collapses the grid to the single comparison the paper is built on. Every one of the 15 cells favors FGPO over GRPO, and the spread is informative rather than uniform: the margin is largest on GenBench-X ($+11.20$ to $+14.20$), where tool evidence is most decisive, and smallest on GenomeQA with the 1.5B reasoner ($+0.92$), where the reasoner is too weak to exploit a better selection at all. In this grid, variation across benchmarks exceeds variation across reasoners in the FGPO–GRPO margin.

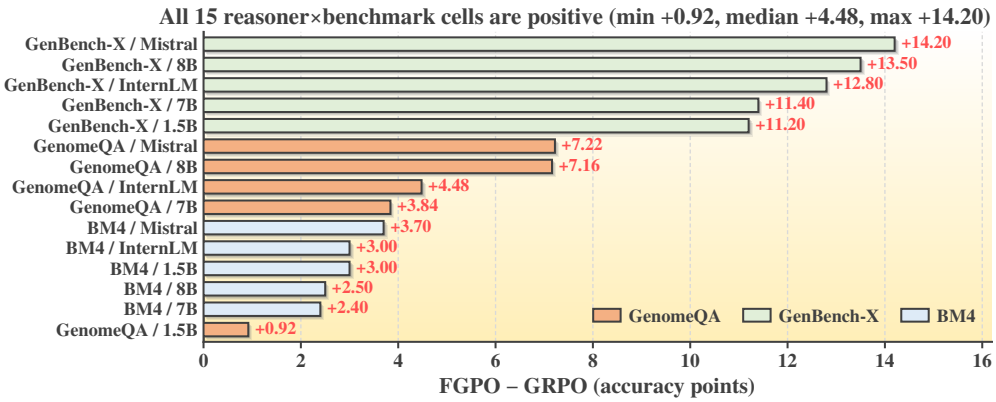


Figure 9: FGPO − GRPO for all 15 reasoner×benchmark cells, sorted. No cell is negative. Colour denotes benchmark.

C.4 Pipeline agreement with the exhaustive table

Every Qwen3-8B number in the paper is read from the exhaustive reward table, the same table training uses; the four transfer reasoners have no such table and are queried live through the transfer pipeline. As a check, the live pipeline re-scored the GenomeQA Qwen3-8B row independently and agrees with the table to within 0.05 points on every row (FGPO 52.65 vs. 52.62; No-Tool 39.33 vs. 39.28; GRPO 45.43 vs. 45.46; All-Tools 49.44 vs. 49.39), i.e. one or two of 3,590 questions, a residual of response nondeterminism at the reasoner server rather than a scoring difference. The reported FGPO checkpoint appears in Tables 3 and 11 at the same table score (52.62); the per-epoch harness that produced their other entries scored a second generation pass of it at 52.59.

C.5 Ceilings: the best fixed subset and the per-question oracle

Two reference lines are computable from the exhaustive tables at no additional cost, and we report them here rather than in the main comparison because neither is a deployable method. BestFixed* applies one subset uniformly to every question: the best of all 16 chosen *post-hoc on the test set* under the training reasoner Qwen3-8B, then transplanted unchanged to the other four reasoners, exactly as the learned policy is. The per-question oracle picks the best subset separately for each question. The oracle upper-bounds every strategy; BestFixed* upper-bounds fixed subsets in the Qwen3-8B column and is a test-label-selected fixed reference elsewhere, which is why it can fall below All-Tools or No-Tool for a reasoner that ranks the subsets differently. Neither is a baseline a practitioner could run.

Table 10 gives BestFixed* against FGPO on every cell. FGPO leads it in 10 of the 15, and the remaining margins are 0.10–2.10: without access to test labels the learned policy closely tracks a reference selected with them. The comparison also shows that the winning subset is not stable: it is T_4 alone on GenBench-X, $T_3{+}T_4$ on BM4 and $T_2{+}T_4$ on GenomeQA, so no single curation choice transfers across benchmarks.

Table 10: FGPO against BestFixed*, the fixed subset selected post-hoc on the test set under Qwen3-8B and transferred to the other reasoners. **Red** marks the larger of the two per cell.

Benchmark		Qwen3-8B	Qwen2.5-1.5B	Qwen2.5-7B	Mistral-7B	InternLM2.5-7B
GenomeQA	FGPO	**52.62**	**43.79**	**50.58**	**42.90**	**52.53**
	BestFixed*	49.97	43.20	47.80	35.79	49.30
GenBench-X	FGPO	65.40	60.00	**68.90**	58.20	**66.90**
	BestFixed*	**67.50**	**60.10**	68.80	**58.80**	66.80
BM4	FGPO	54.80	**51.80**	**55.30**	**48.90**	54.70
	BestFixed*	**55.60**	51.20	50.70	48.80	**54.90**

The per-question oracle sits far above both: 77.41 on GenomeQA under the training reasoner, 90.60 on GenBench-X and 86.90 on BM4, against FGPO's 52.62, 65.40 and 54.80. We read that gap as a statement about the problem rather than about the method (selection over these libraries is nowhere near solved) but it is an upper bound no policy in this paper approaches, and we do not present it as one that any policy should be expected to reach.

C.6 Paired significance

Question-paired McNemar tests, pooled per reasoner ($n \approx 5{,}590$): FGPO vs. GRPO: $p = 2{\times}10^{-8}$ / $8{\times}10^{-19}$ / $2{\times}10^{-39}$ / $2{\times}10^{-30}$ / $1{\times}10^{-20}$ (1.5B/7B/8B/Mistral/InternLM). FGPO vs. Tools w/ Desc: $p = 2{\times}10^{-3}$ / $1{\times}10^{-16}$ / $9{\times}10^{-19}$ / $2{\times}10^{-4}$ / $2{\times}10^{-15}$, a significant win in every column.

D Additional Analyses

D.1 k-subset interpolation in full

Table 3 gives every epoch of every k arm; the main text reports peaks only.

Figure 10 plots the same runs. Coverage does not merely shift the final number: it shifts the whole trajectory, with every arm peaking at the dev-selected epoch 3 and the $k{=}4$ arm collapsing hardest afterwards. The peak values give the dose–response the argument needs (Figure 1c): accuracy rises monotonically in k while the data, ordering, optimization settings and scoring construction are held fixed, so within this construction coverage of the action space is doing the work.

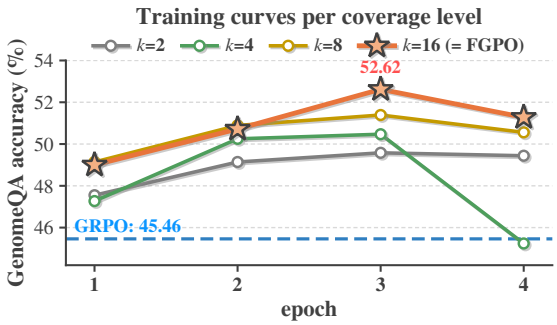


Figure 10: The coverage dial: per-epoch training curves for $k \in \{2, 4, 8, 16\}$ actions evaluated per visit. $k{=}16$ is FGPO itself.

D.2 Per-question paired analysis and selection heatmap

Figure 11 decomposes the aggregate outcome of Figure 6c by task family, and contrasts FGPO's choices with the per-question oracle.

D.3 Data-scale ablation

Table 11 trains FGPO on subsets of the 2,002-question training pool. Accuracy is evaluated on the full 3,590-question GenomeQA test set at every epoch; the best epoch is dev-selected. With only 500 training questions FGPO already reaches 52.23, within 0.4 points of the full-data peak (52.62), and the 1,000-question arm peaks at 50.92. The relationship is not monotone, since 500 slightly outperforms 1,000, but all three are well above GRPO's 45.46.

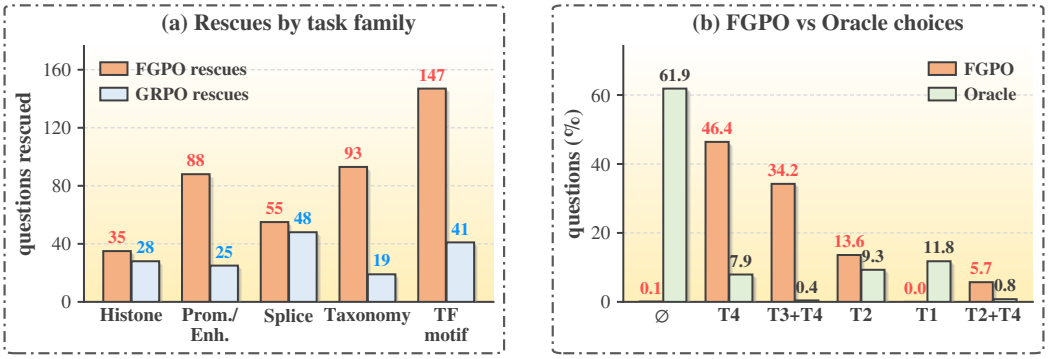


Figure 11: **Per-question analysis on GenomeQA.** (a) Where the 418 FGPO rescues and the 161 GRPO rescues of Figure 6c fall across task families. (b) FGPO vs. the oracle: the oracle prefers the empty set on 61.9% of questions, yet FGPO almost never abstains.

Table 11: Data-scale ablation on GenomeQA. Accuracy (%) at each epoch; peak is dev-selected. All arms use the same hyperparameters.

Training $\lvert\mathcal{D}\rvert$	ep1	ep2	ep3	ep4	peak
500	46.99	50.03	51.14	**52.23**	**52.23**
1,000	49.64	49.22	**50.92**	49.16	50.92
2,002 (full)	49.00	50.72	**52.62**	51.28	**52.62**
GRPO (2,002)			—		45.46

D.4 Effect of parsimony on accuracy and tool usage

Table 12: **Reward design.** The exact objective trained on Eq. 1 (correctness plus the parsimony term) against correctness alone (λ=0), both scored over the enumerated candidates on the 1,200-question quick split. Independent runs with different initialization.

	ep1	ep2	ep3	ep4
With parsimony, acc.	47.25	48.33	**51.17**	49.75
Correctness only, acc.	47.58	48.33	48.75	49.17
With parsimony, tools	1.04	1.53	1.61	1.19
Correctness only, tools	2.04	2.33	2.06	2.19

Penalizing redundancy primarily buys tool economy. Table 12 retrains the exact objective on correctness alone (λ=0 in Eq. 1) against the reported reward, which adds the parsimony term. The λ=0 arm is also the closest exact-objective counterpart of GRPO's reward: on the enumerated candidates Tong et al.'s differential reward is half the ± 1 correctness reward plus a per-question constant, to which the expected-reward term of Eq. 2 is invariant, so that term has a proportional gradient under either; the entropy weight and GRPO's sampling and group normalization lie outside this identity. Between the two arms accuracy differs modestly (49.17 vs. 51.17 at peak) but economy does not: without the term the policy invokes more tools at every epoch (2.0–2.3 against 1.0–1.6; 2.19 vs. 1.19 at epoch 4), as expected when a right answer with four tools scores exactly as well as with one.

D.5 Tool-library single-removal ablations

Table 13 prices each tool by what its removal costs the two ceilings, computed from the exhaustive tables without running a model. No single tool is dispensable on all three benchmarks, but the damage is wildly uneven: removing T_4 costs the GenBench-X oracle 15.5 points and the best fixed subset 20.5, while removing T_3 costs the best fixed subset at most 0.30 anywhere, an option that only pays on questions a fixed subset was never going to get right.

Table 13: Ceiling loss from removing one tool from the full library (computed from the exhaustive tables; no model runs). Δoracle / Δbest-fixed in points.

Benchmark	$-T_1$	$-T_2$	$-T_3$	$-T_4$
GenomeQA	−3.29 / 0.00	−6.21 / −2.20	−3.26 / 0.00	−10.84 / −4.76
GenBench-X	−2.10 / 0.00	−2.70 / 0.00	−1.40 / 0.00	−15.50 / −20.50
BM4	−2.70 / 0.00	−3.10 / 0.00	−1.20 / −0.30	−14.40 / −10.00

E Baseline Fairness Sweeps

No baseline in this paper is reported at a single arbitrary configuration.

E.1 GRPO checkpoint and configuration sweeps

Table 14: GRPO on GenomeQA: all eight checkpoints of the reported run, and four further independent configurations. The main tables report the best checkpoint (step 75, 45.46).

Checkpoint	25	50	75	100	125	150	175	200
Accuracy	42.31	44.85	**45.46**	45.24	44.90	45.32	45.29	45.13
Mean tools	1.33	1.92	2.36	2.18	2.17	2.25	2.15	2.17
Independent configurations (varying G, temperature, data mix): 43.90 / 44.26 / 44.35 / 45.57.								

E.2 DPO β sweep

Table 15: DPO on GenomeQA: every checkpoint of every β. The best observed cell reaches 39.33, essentially the no-tool floor.

β	Checkpoint accuracies	Mean tools	Verdict
0.1	39.25 / 39.28 / 39.25	0.00	collapsed
0.3	39.28 / 39.33	0.03 / 0.01	collapsed
0.5	38.97 / 39.00	0.10 / 0.08	collapsed

Figure 12 shows both sweeps. Panel (a) is the check that matters for the headline comparison: GRPO's accuracy is flat from step 50 onwards while its mean tool count is also flat, so the reported 45.46 is a converged plateau rather than an undertrained checkpoint we happened to stop at. Panel (b) shows the DPO failure has no β that rescues it: every checkpoint of every β lands within 0.4 points of the no-tool floor, because the policy has stopped selecting tools at all.

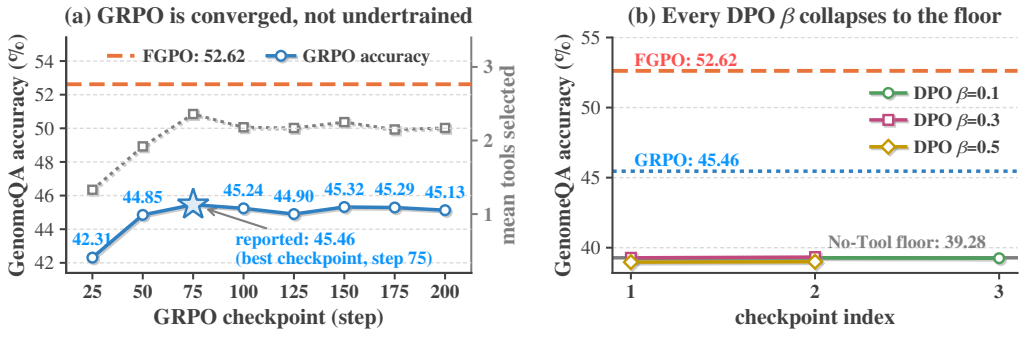


Figure 12: Baseline fairness sweeps on GenomeQA. (a) GRPO across all eight checkpoints, mean tools on the right axis. (b) Every checkpoint of every DPO β.

E.3 GRPO AND DAPO TRAINED ON EQ. 1

The GRPO‡ and DAPO‡ columns of Table 2 separate the estimator from the reward. GRPO‡ is the reported GRPO run with only its reward term replaced: the same configuration file, data, prompts and LoRA adapter, 64 prompts $\times$ G=6 rollouts per step for 200 steps at temperature 1.0, learning rate 10^{-5}, KL coefficient 0.001 and clipping ϵ=0.2, with the reward $0.95\,r + 0.05\,r_{\text{fmt}}$ now taking r from Eq. 1 instead of r_{diff}. DAPO‡ applies the DAPO recipe (Yu et al., 2026) to the same setup: dynamic sampling, which replaces every zero-advantage group with freshly drawn prompts for up to four resampling rounds per generation step; clip-higher (ϵ_{high}=0.28); and no KL term. Token-level loss aggregation is TRL's default and is shared with the reported GRPO, and overlong reward shaping does not apply because completions stay far below the 32-token cap. Both arms are built on TRL's `GRPOTrainer`, checkpointed every 25 steps and selected on the 492-question dev set, the protocol used for SFT.

Table 16 lists all sixteen checkpoints. None escapes the no-tool policy: each invokes at most 0.02 tools per dev question and lies within 0.81 points of No-Tool (36.59), so the collapse does not depend on which checkpoint is reported. Nor is the setup at fault: trained on the differential reward, the identical configuration is the reported GRPO run, which calls 2.36 tools per question at its best checkpoint (Appendix E.1).

Table 16: GRPO‡ and DAPO‡ on the GenomeQA dev set, every checkpoint. No-Tool scores 36.59 on the same questions; the dev-selected checkpoint (step 25 for both) is the one reported in Table 2.

Arm		**25**	**50**	**75**	**100**	**125**	**150**	**175**	**200**
GRPO‡	Accuracy	**37.40**	36.79	36.79	36.79	36.79	36.79	36.59	36.79
	Mean tools	0.02	0.01	0.01	0.01	0.00	0.00	0.00	0.01
DAPO‡	Accuracy	**36.79**	36.59	36.59	36.59	36.59	36.59	36.59	36.59
	Mean tools	0.01	0.00	0.00	0.00	0.00	0.00	0.00	0.00

F MECHANISM DETAILS

F.1 TWO PROPERTIES OF DEAD GROUPS AND OF THE EXACT OBJECTIVE

Fix a question s, its reward table $r(s,\cdot)$ over $\mathcal{A}$, and a sampling distribution p over $\mathcal{A}$ (for the diagnostics of Section 3.4, $p = p_\theta^{\text{diag}}(\cdot \mid s)$). Partition $\mathcal{A}$ into reward classes $\{C_v\}$, one per distinct reward value v, and write $w_v = \sum_{a \in C_v} p(a)$ for the class masses, so $\sum_v w_v = 1$. A group of G i.i.d. draws is dead when all G land in one class, which happens with probability

$$P_{\text{dead}}(p, G) \;=\; \sum_v w_v^G. \tag{7}$$

Proposition 1 (reward collisions). Let $G \geq 2$ be an integer. (i) *Coarsening.* If two reward classes are merged (as happens when a finer reward is replaced by a coarser one that assigns them the same value), P_{dead} does not decrease. (ii) *Concentration.* If mass $0 < \delta \leq w_u$ is moved from a class u to a distinct class v with $w_v \geq w_u$, leaving all other masses fixed, P_{dead} strictly increases. For a fixed number of classes, P_{dead} is Schur-convex in $(w_v)_v$, so it is minimized at the uniform class distribution and maximized when one class carries all the mass.

Proof. (i) For $a, b \geq 0$ and $G \geq 2$, $(a+b)^G = \sum_{j=0}^{G} \binom{G}{j} a^j b^{G-j} \geq a^G + b^G$, so replacing the two terms $w_u^G + w_v^G$ in Eq. 7 by $(w_u + w_v)^G$ cannot lower the sum. (ii) The function $f(x) = x^G$ is strictly convex on $[0, \infty)$ for $G \geq 2$, so $f(w_v + \delta) + f(w_u - \delta) - f(w_v) - f(w_u) = \int_0^\delta \left[f'(w_v + t) - f'(w_u - t)\right] dt > 0$ because f' is strictly increasing and $w_v + t > w_u - t$ for all $t \in (0, \delta]$. Schur-convexity follows because $\sum_v f(w_v)$ with f convex is Schur-convex (Marshall et al., 1979). □

Proposition 1 gives conditions for interpreting the diagnostics of Section 3.4. Under the same sampling distribution, part (i) orders the 20.8% closed-form rate under the fine partition of Eq. 1 below the 79.6% rate under the coarser differential reward. Part (ii) concerns concentration of *reward-class*

mass; a decrease in policy entropy alone does not establish this condition. The concurrent entropy decrease and dead-rate increase in Figure 7 are consistent with this mechanism, but do not establish a majorization ordering of the class-mass vectors. The propositions apply regardless of whether the policy is correct; the observed near-independence of dead mass and correctness is an empirical finding of Section 3.4, not a consequence of these properties.

Proposition 2 (the exact objective). Fix s and θ, and let $q_\theta(\cdot \mid s)$ be the softmax in Eq. 4 with finite logits $z_a = \ell_\theta(a \mid s)$. Let $J_s(\theta) = \sum_a q_\theta(a \mid s)\, r(s,a)$ be the per-question objective of Eq. 2. In exact arithmetic, (i) J_s and $\nabla_\theta J_s$ are computed exactly from the $|\mathcal{A}|$ candidate scores, so the estimator has zero variance with respect to action sampling. Random minibatches can still introduce variance across questions. (ii) The reward-term gradient with respect to the logit of action a is

$$\frac{\partial J_s}{\partial z_a} \;=\; q_\theta(a \mid s)\,\big(r(s,a) - J_s\big), \tag{8}$$

Since finite softmax logits give $q_\theta(a \mid s) > 0$, every action with $r(s,a) \neq J_s$ has a non-zero reward-term logit gradient whose sign is that of its advantage over the current expected reward, and whose magnitude is proportional to $q_\theta(a \mid s)$.

Proof. (i) is immediate from Eq. 2 being a finite sum with no sampled term. (ii) With $q_a = \exp z_a / \sum_{a'} \exp z_{a'}$, $\partial q_b / \partial z_a = q_b(\mathbf{1}[a{=}b] - q_a)$, hence $\partial J_s / \partial z_a = \sum_b r_b\, q_b(\mathbf{1}[a{=}b] - q_a) = q_a r_a - q_a \sum_b q_b r_b = q_a(r_a - J_s)$. □

These statements concern the reward term in logit space. Exhaustive evaluation rules out zeros caused by sampled reward collisions, but small $q_\theta(a \mid s)$ can still make the logit gradient arbitrarily small. LoRA parameter updates couple candidate logits, so a non-zero logit derivative guarantees neither an increase in that action's probability nor fast escape. The entropy bonus of Eq. 5 adds a separate gradient and encourages dispersion; it guarantees neither a probability floor during training nor rapid recovery. The k-subset intervention of Section 3.5 changes action coverage and renormalizes q_θ within each sampled subset, thereby changing the objective; its gradient need not be unbiased for J_s. At $k = |\mathcal{A}|$, it recovers the full exact objective.

F.2 Dead-group grids for all three benchmarks

Table 17 gives the closed-form rates behind Figure 5 at every group size we evaluated, for all four exported distributions rather than the two the main text plots.

How p_θ^{diag} is obtained. For each test question we append each of the 16 legal answer strings (e.g. `<answer>0,2</answer>`) and the end-of-sequence token to the policy prompt, sum their token log-probabilities under the policy by teacher forcing (temperature 1, no length normalization), and take a softmax over the 16 sums. It approximates p_θ^{gen}, temperature-1 free generation, conditioned on emitting one of the legal answers. No sampling is involved: every rate is computed from these 16 probabilities and the cached rewards on all test questions (3,590 / 1,000 / 1,000). GRPO's greedy completions are well-formed on every test question of all three benchmarks, and the live telemetry of Section 3.4 counts dead groups directly under temperature-1.0 sampling, malformed completions included.

Read across the three grids, the two curves that matter run in opposite directions and neither is an accident of tuning: a uniform reference policy is almost never dead beyond $G{=}4$, and under the fine-grained Eq. 1 partition the converged GRPO policy has a 17–21% dead-group probability at the $G{=}6$ group size used for training. FGPO's own row is included precisely because it is the worst of the four: its converged distribution would be nearly useless to sample from (94% dead at $G{=}6$), which is the point, since it never samples. SFT's low dead rate is not evidence of useful routing: its sampling distribution stays relatively diffuse, but its argmax collapses onto the empty action on 92.8% of questions, and its accuracy (42.17) trails GRPO's regardless.

F.3 Empty-subset preference: how offline objectives collapse

SFT's high mass on the modal action also explains its deceptively favorable "chance-of-sampling-an-optimal-action" statistics: mimicking the mode looks good on any metric that ignores the other 38% of questions.

Table 17: Closed-form dead-group rate (%) vs. group size G, per benchmark, the numerical form of Figure 5a–c, computed from each policy's p_θ^{diag}, its generation distribution conditioned on the 16 valid subset strings. Reward classes follow Eq. 1, the finest partition any arm induces. SFT, GRPO and FGPO are reported on all three benchmarks.

Benchmark	Policy	G=1	G=2	G=4	G=6	G=8	G=16	G=32
GenomeQA	Uniform	100.0	22.7	1.9	0.2	0.0	0.0	0.0
	SFT	100.0	40.4	9.1	2.4	0.8	0.0	0.0
	GRPO	100.0	56.5	30.4	**20.8**	15.7	7.7	3.7
	FGPO	100.0	97.3	95.1	94.0	93.2	91.5	89.7
GenBench-X	Uniform	100.0	20.4	1.4	0.1	0.0	0.0	0.0
	SFT	100.0	37.6	7.5	1.7	0.4	0.0	0.0
	GRPO	100.0	53.5	26.9	**17.8**	13.4	6.2	1.8
	FGPO	100.0	97.1	94.7	93.4	92.5	90.5	88.7
BM4	Uniform	100.0	20.5	1.5	0.1	0.0	0.0	0.0
	SFT	100.0	38.2	7.8	1.9	0.5	0.0	0.0
	GRPO	100.0	56.1	29.0	**17.4**	11.1	2.9	0.8
	FGPO	100.0	98.4	97.2	96.6	96.2	95.4	94.7

Table 18: Generation-induced probability mass and argmax rate on the empty subset (no tools), per trained policy, over the GenomeQA test set; both columns are computed from p_θ^{diag}, the same distribution as Table 17. Reference: the empty subset is reward-optimal on 61.9% of questions.

Policy	Generation-induced mass on $\emptyset$	Argmax $= \emptyset$	GenomeQA accuracy
SFT	42.2%	92.8%	42.17
GRPO	4.0%	2.8%	45.46
FGPO	0.2%	0.2%	52.62

Figure 13 shows the mechanism behind the offline collapse: the empty subset is reward-optimal on 61.9% of questions, SFT puts 92.8% of its argmax there, and the resulting policy is a very good imitation of the modal answer and a very poor policy.

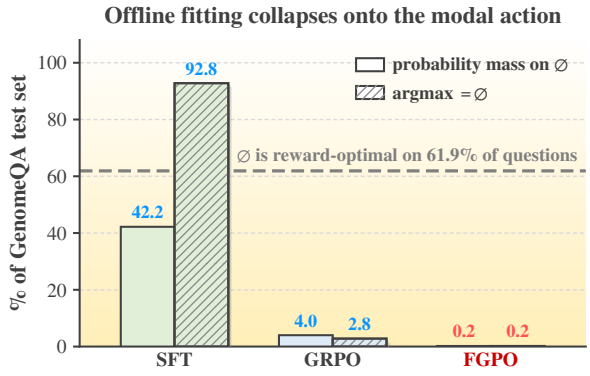


Figure 13: Generation-induced probability mass and argmax rate on the empty subset per trained policy. The empty subset is reward-optimal on 61.9% of GenomeQA test questions (dashed line).

G Prompt Templates

G.1 Policy prompt (all trained arms: FGPO, GRPO, SFT, DPO)

Tools are anonymous slots; the policy answers in indices. System:

```
You are an expert agent specialized in selecting tools to solve
genomic reasoning tasks.  You are provided with access to {n} tools,
indexed from 0 to {n-1}.
```

Tool list and instruction:

```
0:  type1 (A)
1:  type2 (B)
2:  type3 (C)
3:  type4 (D)
{question and options}
Select the index number(s) of the tools that are most helpful for
solving the task.  You MUST output only the selected tool indices as
a comma-separated list, enclosed in <answer></answer> tags.  Output
an empty <answer></answer> if no tool is needed.
```

An empty <answer></answer> selects no tools; our action space includes it explicitly.

G.2 Prompted baseline (Tools w/ Desc)

Identical layout, but each slot carries the tool's real name and a functional description:

```
{i}:  {tool name} - Description:  {functional description}
```

The prompted baseline therefore receives strictly more prior information than any trained policy in this paper.

G.3 Reasoner prompt

The frozen reasoner receives the question, the options, and the rendered evidence of the selected subset (one titled block per tool, shared verbatim across all methods and table rows), and is instructed to answer with a single option letter.

G.4 Complete reasoner prompt

The frozen reasoner receives the following system message and user prompt. The evidence block is empty in the No-Tool arm and contains one titled block per selected tool in all other arms. The same template is shared across every method and every table row; only the evidence differs.

System message.

```
You are a genomics expert answering multiple-choice questions about
DNA sequences.  Tool outputs, when provided, may be incomplete or
noisy.  Use only the evidence that is relevant to the question and
ignore the rest.  Do not assume a tool is correct.  Respond with the
single letter of the best option and nothing else.
```

User message template.

```
{question}
Options:
{options_block}
Analysis tool output for the sequence above:
{evidence}
Answer with a single letter.
```

When no tools are selected, the evidence block is omitted entirely.

G.5 Complete tool evidence examples

Below are schematic evidence blocks condensed from the exhaustive reward tables; the prompt itself carries each tool's JSON output in full. Each example illustrates the output format of one tool; the reasoner sees the concatenation of the selected tools' blocks.

T_1 **(Sequence Composition).**

```
Sequence A: length=371bp, GC=59.3%, AT=40.7%, CpG O/E=0.264,
GC_skew=0.051
Top 3-mers:  TTT (12, 3.3%), TAT (9, 2.4%), TTA (8, 2.2%)
Sequence B: length=371bp, GC=42.1%, AT=57.9%, CpG O/E=0.891,
GC_skew=-0.032
Top 3-mers:  AGG (11, 3.0%), GGA (9, 2.4%), CAG (8, 2.2%)
...(C, D analogous)
```

T_2 **(Motif Scanner).**

```
Sequence A: JASPAR hits (p<1e-4):  JUN (score=12.3), FOS (11.8),
TEAD4 (10.2)
Sequence B: STAT1 (score=14.1), IRF1 (9.7)
Sequence C: SP1 (score=13.5), KLF4 (12.1)
Sequence D: SRF (score=15.8), MEF2A (11.2)
```

T_3 **(Splice-Site Analysis).**

```
Sequence A: no GT/AG sites above threshold (donor>6.0, acceptor>6.0)
Sequence B: acceptor at pos 142 (score=8.53, Only Acceptor)
Sequence C: no sites above threshold
Sequence D: no sites above threshold
```

T_4 **(Genomic Expert).**

```
kNN prediction (k=15, cosine over frozen NT-v2-50m embeddings;
confidence is the
distance-weighted vote share, counts are raw neighbours):
Sequence A: Eukaryote 0.82 (12/15 neighbours)
Sequence B: Virus 0.37 (6/15 neighbours; Eukaryote 0.33, Prokaryote
0.30)
Sequence C: Eukaryote 0.53 (8/15)
Sequence D: Eukaryote 0.84 (13/15)
```

T_5 **(Homology Search), used only in the five-tool stress test of Appendix B.3.**

```
Sequence A: 2 hits.  ref_04188 (label=promoter, identity=71.4%,
coverage=58%);
            ref_11902 (label=enhancer, identity=64.0%,
coverage=41%).
            Consensus:  promoter (1/2, agreement 0.50)
Sequence B: no hit above threshold (identity ≥ 60%, coverage ≥ 30%)
Sequence C: 1 hit.  ref_00734 (label=enhancer, identity=62.8%,
coverage=35%).
            Consensus:  enhancer (1/1, agreement 1.00)
Sequence D: no hit above threshold
```

Two of the four candidates return nothing; among the remaining two, one has only a $1/2$ consensus and the other rests on a single supporting hit. Evidence of this shape is why the exact objective settles on invoking T_5 for only 2.8% of questions (Appendix B.3): the tool is well implemented but rarely decisive on this benchmark, and the reward makes that visible without any hand-written rule.

G.6 Question format examples

Table 19 shows one verbatim question from each benchmark, illustrating the diversity of task types, option formats and sequence lengths. Every option list is reproduced in full; only the nucleotide sequence is elided in the middle.

Table 19: **Question format across the three benchmarks.** One row per benchmark, quoted verbatim from the benchmark files that we will release. Green marks the gold option. Sequences are shown head . . . tail; the full length is given in the header row.

Question	Options
GenomeQA — TF motif, `describe` shape, 100 bp	
"Identify the transcription factor that binds to this Human DNA sequence: CCCTACAGAG...CTCGAGGCTC"	A: NRF1 B: POU5F1 C: E2F1 **D: RELA**
GenBench-X — taxonomy, `describe` shape, 200 bp	
"Which organism is this DNA sequence from?" CCCAGCCCTA...GTCAGACAGT	A: Caenorhabditis elegans (worm) **B: Human**
BM4 — bacterial promoter, `describe` shape, 81 bp	
"Is this *Escherichia coli* DNA sequence a sigma70 promoter?" TCTACGCCGC...AGGTAGAAGT	**A: Not a promoter** B: Promoter

G.7 Tool description prompt (Tools w/ Desc baseline)

The prompted baseline receives the same layout as the trained policies, but each tool slot carries the real name and a functional description:

```
0: sequence_statistics - Computes GC content, CpG density, k-mer
frequencies, and other compositional statistics of the DNA sequence.
1: motif_scanner - Scans the sequence against JASPAR transcription
factor binding profiles and returns significant hits with their
scores.
2: splice_analyzer - Identifies candidate GT/AG splice sites and
scores them with MaxEntScan donor and acceptor models.
3: genomic_expert - Classifies the sequence by kNN vote over frozen
Nucleotide-Transformer embeddings of a labelled reference set.
```

The prompted baseline therefore receives strictly more prior information than any trained policy in this paper. Its accuracy (47.74 on GenomeQA) exceeds GRPO (45.46), showing that access to tool semantics is valuable, but FGPO surpasses both (52.62) without any semantic information.

H Additional Case Studies

Table 7 showed three rescues. This section adds seventeen more, drawn by a fixed rule rather than chosen by hand. For each benchmark we take the test questions that the frozen Qwen3-8B reasoner answers wrongly without tools and correctly with FGPO's selection, with FGPO calling at least one tool; Table 23 counts them. Within each benchmark and task slot (for GenBench-X and BM4 each slot holds one source subset, Appendix B.8) we keep the questions on which the reported GRPO policy is also wrong and draw one uniformly at random (a single `random.Random(20260924)` stream, families in alphabetical order, candidates sorted by question id), then draw one further question per benchmark for a fully expanded walk-through. Selections, answers and tool outputs are read from the evaluation records behind Table 2. DNA sequences are shortened to their first and last ten bases except in the walk-throughs, and tool outputs are the JSON the reasoner receives, re-indented for space, with two edits: the local path of the frozen embedding weights is redacted, and the implementation's slot name `T5` is printed as T_4. The draws are not filtered for how clear-cut the evidence looks: in several of them the tool output favours the correct option only weakly, and the reasoner still has to weigh it.

H.1 One rescue per benchmark and task family

Table 20: **GenomeQA rescues**, one per task family, drawn by the rule above. Answers are Qwen3-8B's; wrong in red, correct in green. Tool lines condense FGPO's tool outputs (per option where the tool ran per option); the named factor is in bold, kNN confidences in parentheses.

No-Tool / GRPO	FGPO: selection and tool evidence	FGPO
Histone mark (`bcq_4_histone_marks_135`): "I'm analyzing ChIP-seq data from K562 cells. Is the DNA sequence `TGGGGATGCC`…`TTCAAGACCA` (1,000 bp) enriched for H3K27ac?" Options: A. Yes; B. No. [GT: B]		
No-Tool: A GRPO (T_4): A	FGPO calls T_3+T_4. T_3 splice pattern: no site at the candidate position, only background hits elsewhere T_4 kNN prediction: H3K9ac (0.25)	B
Promoter/enhancer (`mcq_1_enhancer_vs_promoter_498`): "I need to distinguish the specific sequence that contains a promoter from the others that carry enhancer. Which one is it?" Options: four 400-bp sequences. [GT: D]		
No-Tool: B GRPO (T_1+T_3+T_4): A	FGPO calls T_4. T_4 kNN prediction: A: Enhancer (0.57) | B: Not an enhancer (0.26) | C: Enhancer (0.41) | D: Enhancer (0.58)	D
Splice site (`mcq_2_splice_sites_133`): "Classify the splicing function of this Human DNA sequence: `GTTCCTGGGG`…`TTGGTGCAAA` (600 bp)." Options: A. Neither (No Splice Site); B. Only Acceptor; C. Both Acceptor and Donor; D. Only Donor. [GT: D]		
No-Tool: C GRPO (T_3+T_4): A	FGPO calls T_4. T_4 kNN prediction: Only Donor (0.54)	D
Taxonomy (`bcq_3_species_cls_443`): "Does the DNA sequence `GAAATTTGTA`…`TTGATGATTT` (1,000 bp) belong to the Virus species?" Options: A. No; B. Yes. [GT: B]		
No-Tool: A GRPO (T_4): A	FGPO calls T_3+T_4. T_3 splice pattern: no site at the candidate position, only background hits elsewhere T_4 kNN prediction: Virus (0.41)	B
TF motif (`mcq_5_tfbs_68`): "Select the Human DNA sequence that is bound by the transcription factor FOXA1." Options: four 100-bp sequences. [GT: C]		
No-Tool: D GRPO (T_3+T_4): no letter	FGPO calls T_2. T_2 motif hits: A: REST | B: ATF3, USF1, SRF | C: **FOXA1**, STAT1 | D: REST, GATA1	C

Table 21: **GenBench-X rescues**, one per source subset, labelled with the task slot it fills (Appendix B.8); layout as in Table 20.

No-Tool / GRPO	FGPO: selection and tool evidence	FGPO
human_ocr_ensembl (histone-mark slot, gb2_human_ocr_en_sel_0010): "Which of the following Human DNA sequences lies in an open chromatin region?" Options: four sequences (417, 273, 204, 342 bp). [GT: B]		
No-Tool: D GRPO (T_2+T_3+T_4): A	FGPO calls T_4. T_4 kNN prediction: A: No (0.68) | B: Yes (0.80) | C: Yes (0.66) | D: No (0.50)	B
human_ensembl_regulatory (promoter/enhancer slot, gb2_human_ensemb_1_0041): "Which regulatory element does this Human DNA sequence represent?" Options: A. Enhancer; B. Open chromatin region; C. Promoter. [GT: B]		
No-Tool: A GRPO (T_1+T_2+T_3): A	FGPO calls T_4. T_4 kNN prediction: Open chromatin region (0.73)	B
demo_coding_vs_intergenomic (splice-site slot, gb2_demo_coding__sel_0055): "Which of the following DNA sequences is a Intergenomic sequence?" Options: four 200-bp sequences. [GT: B]		
No-Tool: D GRPO (T_2+T_3+T_4): D	FGPO calls T_4. T_4 kNN prediction: A: Coding sequence (0.67) | B: Intergenomic sequence (0.94) | C: Coding sequence (1.00) | D: Intergenomic sequence (0.60)	B
demo_human_or_worm (taxonomy slot, gb2_demo_human_o_1_0069): "Which organism is this DNA sequence from?" Options: A. Caenorhabditis elegans (worm); B. Human. [GT: B]		
No-Tool: A GRPO (T_1+T_2): A	FGPO calls T_3+T_4. T_3 splice pattern: no splice signal anywhere in the window T_4 kNN prediction: Human (0.63)	B
drosophila_enhancers_stark (TF-motif slot, gb2_drosophila_e_sel_0032): "Which of the following Drosophila DNA sequences functions as an enhancer?" Options: four 1000-bp sequences. [GT: B]		
No-Tool: C GRPO (T_2+T_3+T_4): A	FGPO calls T_4. T_4 kNN prediction: A: Yes (0.59) | B: Yes (0.65) | C: No (0.75) | D: No (0.67)	B

Table 22: **BM4 rescues**, one per source subset, labelled with the task slot it fills (Appendix B.8); layout as in Table 20.

No-Tool / GRPO	FGPO: selection and tool evidence	FGPO
Mouse Ensembl enhancers (histone-mark slot, `bm4_mouse_sel_0062`): "Which of the following Mus musculus DNA sequences is an Not an enhancer?" Options: four 1000-bp sequences. [GT: B]		
No-Tool: C GRPO (T_2+T_3+T_4): A	FGPO calls T_4. T_4 kNN prediction: A: Enhancer (0.72) | B: Not an enhancer (0.80) | C: Enhancer (1.00) | D: Not an enhancer (0.67)	B
Bacterial σ^{70} promoters (promoter/enhancer slot, `bm4_sigma70_sel_0059`): "Which of the following Escherichia coli DNA sequences is a Promoter?" Options: four 81-bp sequences. [GT: A]		
No-Tool: C GRPO (T_2): B	FGPO calls T_4. T_4 kNN prediction: A: Promoter (0.60) | B: Not a promoter (0.59) | C: Not a promoter (0.54) | D: Not a promoter (0.62)	A
Bacteria vs. archaea (taxonomy slot, `bm4_domain_1_0026`): "Which domain of life is this DNA sequence from?" Options: A. Archaea; B. Bacteria. [GT: B]		
No-Tool: A GRPO (T_1+T_2): A	FGPO calls T_3+T_4. T_3 splice pattern: no site at the candidate position, only background hits elsewhere T_4 kNN prediction: Bacteria (0.55)	B
DeepSTARR fly enhancers (TF-motif slot, `bm4_starr_1_0007`): "Does this Drosophila enhancer drive developmental or housekeeping activity?" Options: A. Developmental enhancer; B. Housekeeping enhancer. [GT: B]		
No-Tool: A GRPO (T_1+T_2): A	FGPO calls T_2+T_4. T_2 motif hits: JUN, FOS T_4 kNN prediction: Housekeeping enhancer (0.73)	B

Table 23: Size of the rescue pool: test questions answered wrongly by Qwen3-8B without tools and correctly under FGPO's selection (FGPO calling at least one tool). In parentheses, those on which the reported GRPO is also wrong, the pool the cases above are drawn from. Columns are GenomeQA's task families, which for GenBench-X and BM4 are the task slots their source subsets fill (Appendix B.8). Benchmarks have 3,590 / 1,000 / 1,000 questions; no BM4 source fills the splice-site slot.

Benchmark	Histone	Prom./enh.	Splice	Taxonomy	TF motif	Total
GenomeQA	115 (27)	209 (73)	102 (33)	212 (69)	241 (110)	879 (312)
GenBench-X	90 (7)	88 (55)	73 (6)	80 (64)	74 (16)	405 (148)
BM4	106 (10)	61 (18)	—	66 (23)	77 (19)	310 (70)

H.2 Walk-through: GenomeQA, TF motif

Question (`mcq_5_tfbs_102`). "Select the DNA sequence containing the motif for the master regulator of chromatin structure (CTCF)."

Option A (100 bp):
`CACGGAGAGCAGCTGTTGTGGGCGCCCCACCCACGCCTCCAACCCAAGGGCTTTCTCAAG`
`CCATCCCACTGTGCTCGGAGACGTAAGTGGGTTCTAGAAG`
Option B (100 bp):
`GTAGGAAATTCCACTGCATAATTTGTGGTAGTGGGGGGGGCTGCGTTAGTGAATTCCGCT`
`GAGATACTGCAGTGCAAAACGTGAGCTCTACGCGAAGGGC`
Option C (100 bp):
`TCTGACACTTTCTTGGTAGTTCCCTGCGTCCCCAAGTAAGCCTGCGCAGACTGGGAAGTC`
`TGGGAAGTCGGAGAAGGAAGGCGCTACCAAGCCGGCAACG`
Option D (100 bp):
`ACCGCGTCGAGGCAGCGCGAGCCCCGGGCGCCCCCTGCAGGTTGTACGGGCCTCAGCTGC`
`GCAGCGCAGACGGAGGAGGGCGCTGGCGGTTAAGAACTCC`

Outcome. Ground truth: D. Without tools Qwen3-8B answers C. GRPO emits <answer>2</answer>, i.e. T_3, and the reasoner answers C. FGPO emits <answer>1</answer>, i.e. T_2, and the reasoner answers D. The table below is this question's row of the exhaustive reward table: the reasoner's answer under every legal subset, with FGPO's choice marked * and GRPO's $^\circ$; subsets are written by tool index (1+3 is T_1+T_3), and – marks a reply with no parseable letter.

Subset	none	1	2*	1+2	3°	1+3	2+3	1+2+3
Answer	C	A	D	D	C	A	D	D
Subset	4	1+4	2+4	1+2+4	3+4	1+3+4	2+3+4	1+2+3+4
Answer	D	A	D	D	–	A	D	D

Tool output passed to the reasoner under FGPO's selection.

Option A:

```
[T2] motif_scanner:
{
  "backend": "biopython",
  "database": "JASPAR2024",
  "factors_named_in_question": ["CTCF"],
  "n_motifs_scanned": 32,
  "p_value_threshold": 0.0001,
  "n_significant_hits": 1,
  "matched_transcription_factors": ["EGR1"],
  "hits_per_factor": {"EGR1": 1},
  "hits": [
    {"motif": "EGR1", "position": 26, "strand": "+", "score": 14.072, "relative_score": 0.843,
        "p_value": 6.676e-06, "matched_sequence": "CCACCCACGC"}
  ]
}
```

Option B:

```
[T2] motif_scanner:
{
  "backend": "biopython",
  "database": "JASPAR2024",
  "factors_named_in_question": ["CTCF"],
  "n_motifs_scanned": 32,
  "p_value_threshold": 0.0001,
  "n_significant_hits": 2,
  "matched_transcription_factors": ["RELA", "TEAD4"],
  "hits_per_factor": {"RELA": 1, "TEAD4": 1},
  "hits": [
    {"motif": "RELA", "position": 3, "strand": "-", "score": 14.983, "relative_score": 0.893,
        "p_value": 5.722e-06, "matched_sequence": "GGAAATTCCA"},
    {"motif": "TEAD4", "position": 5, "strand": "+", "score": 11.543, "relative_score": 0.786,
        "p_value": 6.104e-05, "matched_sequence": "AAATTCCA"}
  ]
}
```

Option C:

```
[T2] motif_scanner:
{
  "backend": "biopython",
  "database": "JASPAR2024",
  "factors_named_in_question": ["CTCF"],
  "n_motifs_scanned": 32,
  "p_value_threshold": 0.0001,
  "n_significant_hits": 0,
  "matched_transcription_factors": [],
  "hits_per_factor": {},
  "hits": []
}
```

Option D:

```
[T2] motif_scanner:
{
  "backend": "biopython",
  "database": "JASPAR2024",
  "factors_named_in_question": ["CTCF"],
  "n_motifs_scanned": 32,
  "p_value_threshold": 0.0001,
  "n_significant_hits": 1,
  "matched_transcription_factors": ["CTCF"],
  "hits_per_factor": {"CTCF": 1},
  "hits": [
    {"motif": "CTCF", "position": 27, "strand": "-", "score": 13.622, "relative_score": 0.834,
        "p_value": 1.335e-05, "matched_sequence": "GCGCCCCCTG"}
  ]
}
```

H.3 Walk-through: GenBench-X, human_ensembl_regulatory

Question (gb2_human_ensemb_1_0012). "Which regulatory element does this Human DNA sequence represent?"

Options: A. Enhancer; B. Open chromatin region; C. Promoter.

> Sequence (458 bp):
> AGGGGAAAAATCCATAGGGACATTAGGGCGTTTAAATGTGCTTCTGAAAATGGTTGGACA
> AACTGTAGCCTTGGCACATGAAATAGCGTTGCTCAACCTAAGCAAATGGCTTTGACTGTC
> AGATCAAAACAATTCATTTTATGCAGTTCATATCACAGGCAAGTATTTCTGTAGTCTTAT
> GACTTGTGAAAAACACTAACTAGAAAATGTATGTAACTCCCAATGACTCACTAGAAGTTA
> TGGTATATTTCTACTTTGGCCTGACGGGCATTTCTATTACTCAGCGGATCTGTTATTAGT
> TGTTGTCTAGACGGTCCATTACTCAAGTCTTACGTGTGTTTCATATTCTGAGAGTGTTTC
> ATATTCAGTGAGTAGGCCTGTCAGATGAAATAAAATTGGATGGATGTTTAGTATGAAAAG
> GATTGAGTTTGATAGAAAGAAAAAAACATGTAACTATG

Outcome. Ground truth: B (Open chromatin region). Without tools Qwen3-8B answers A. GRPO emits <answer>0,1,2</answer>, i.e. T_1+T_2+T_3, and the reasoner answers A. FGPO emits <answer>1,3</answer>, i.e. T_2+T_4, and the reasoner answers B. The table below is this question's row of the exhaustive reward table: the reasoner's answer under every legal subset, with FGPO's choice marked * and GRPO's $^\circ$; subsets are written by tool index (1+3 is T_1+T_3), and – marks a reply with no parseable letter.

Subset	none	1	2	1+2	3	1+3	2+3	1+2+3°
Answer	A	C	A	C	A	C	A	A
Subset	4	1+4	2+4*	1+2+4	3+4	1+3+4	2+3+4	1+2+3+4
Answer	B	B	B	A	B	C	B	A

Tool output passed to the reasoner under FGPO's selection.


```
[T2] motif_scanner:
{
  "backend": "biopython",
  "database": "JASPAR2024",
  "factors_named_in_question": null,
  "n_motifs_scanned": 32,
  "p_value_threshold": 0.0001,
  "n_significant_hits": 5,
  "matched_transcription_factors": ["JUN", "FOS", "YY1"],
  "hits_per_factor": {"JUN": 2, "FOS": 2, "YY1": 1},
  "hits": [
    {"motif": "JUN", "position": 223, "strand": "+", "score": 12.469, "relative_score": 0.858,
        "p_value": 3.052e-05, "matched_sequence": "TGACTCAC"},
    {"motif": "FOS", "position": 222, "strand": "-", "score": 14.059, "relative_score": 0.977,
        "p_value": 3.052e-05, "matched_sequence": "ATGACTCA"},
    {"motif": "FOS", "position": 223, "strand": "+", "score": 12.485, "relative_score": 0.868,
        "p_value": 4.578e-05, "matched_sequence": "TGACTCAC"},
    {"motif": "YY1", "position": 45, "strand": "+", "score": 11.474, "relative_score": 0.784,
        "p_value": 4.578e-05, "matched_sequence": "GAAAATGG"},
    {"motif": "JUN", "position": 222, "strand": "-", "score": 12.289, "relative_score": 0.845,
        "p_value": 6.104e-05, "matched_sequence": "ATGACTCA"}
  ]
}
[T4] genomic_expert:
{
  "backbone": "[path redacted]",
  "backbone_frozen": true,
  "method": "cosine kNN (k=15) over frozen embeddings, self-hits excluded",
  "task": "promoter_enhancer",
  "prediction": "Open chromatin region",
  "confidence": 0.5194,
  "class_probabilities": {"Open chromatin region": 0.5194, "Enhancer": 0.355, "Promoter": 0.1256},
  "nearest_neighbour_labels": ["Enhancer", "Enhancer", "Enhancer", "Open chromatin region", "Open
      chromatin region", "Open chromatin region", "Promoter", "Open chromatin region", "Open chromatin
      region", "Enhancer"],
  "mean_neighbour_distance": 0.0113,
  "n_self_hits_excluded": 0
}
```

H.4 Walk-through: BM4, bacterial σ^{70} promoters

Question (`bm4_sigma70_1_0041`). "Is this Escherichia coli DNA sequence a sigma70 promoter?"

Options: A. Not a promoter; B. Promoter.

> Sequence (81 bp):
> `ATGTGATGATCAATTTTACCTTATGGTTAACAGTCTGTTTCGGTGGTAAGTTCAGGCAAA`
> `AGAGAACGATTGCGTTGGGGA`

Outcome. Ground truth: B (Promoter). Without tools Qwen3-8B answers A. GRPO emits `<answer>1,2</answer>`, i.e. T_2+T_3, and the reasoner answers A. FGPO emits `<answer>2,3</answer>`, i.e. T_3+T_4, and the reasoner answers B. The table below is this question's row of the exhaustive reward table: the reasoner's answer under every legal subset, with FGPO's choice marked * and GRPO's $^\circ$; subsets are written by tool index (1+3 is T_1+T_3), and – marks a reply with no parseable letter.

Subset	none	1	2	1+2	3	1+3	2+3°	1+2+3
Answer	A	A	A	A	A	A	A	A
Subset	4	1+4	2+4	1+2+4	3+4*	1+3+4	2+3+4	1+2+3+4
Answer	B	B	B	B	B	B	B	B

Tool output passed to the reasoner under FGPO's selection.

```
[T3] splice_analyzer:
{
  "method": "MaxEntScan",
  "candidate_position": 40,
  "centre_window_bp": 10,
  "donor_threshold_bits": 6.0,
  "acceptor_threshold_bits": 6.0,
  "donor_at_candidate_position": {"found": true, "position": 45, "score_bits": 10.355, "context":
      "GTGGTAAGT"},
  "acceptor_at_candidate_position": {"found": false},
  "overall_pattern": "donor at the candidate position",
  "best_donor_anywhere": {"position": 45, "score_bits": 10.355, "context": "GTGGTAAGT"},
  "best_acceptor_anywhere": null,
  "background_donor_hits": 0,
  "background_acceptor_hits": 0,
  "interpretation": "A strong MaxEntScan score at the candidate position is the evidence that matters;
      hits elsewhere in the window are background. Donor scores near 8-11 bits are strong, 6-8
      marginal."
}
[T4] genomic_expert:
{
  "backbone": "[path redacted]",
  "backbone_frozen": true,
  "method": "cosine kNN (k=15) over frozen embeddings, self-hits excluded",
  "task": "promoter_enhancer",
  "prediction": "Promoter",
  "confidence": 0.6021,
  "class_probabilities": {"Promoter": 0.6021, "Not a promoter": 0.3979},
  "nearest_neighbour_labels": ["Promoter", "Not a promoter", "Not a promoter", "Promoter", "Promoter",
      "Promoter", "Promoter", "Promoter", "Promoter", "Not a promoter"],
  "mean_neighbour_distance": 0.0222,
  "n_self_hits_excluded": 0
}
```

I Further Examples

This section makes the aggregate results concrete with three further kinds of examples and one more walk-through, read from the same evaluation records as Appendix H and drawn by a fixed rule from a separate stream (`random.Random(20260925)`; benchmarks in the order GenomeQA, GenBench-X, BM4; task slots in alphabetical order; candidates sorted by question id). Figure 14 draws four questions per benchmark uniformly from those on which the tool subset matters, i.e. some legal subset makes the reasoner right and some makes it wrong, the complement of the outcome-degenerate questions of Section 3.4; nothing about either policy's choice enters that pool. Appendix I.2 draws one question per task slot from those the reasoner answers correctly both without tools and under FGPO's selection, but wrongly under GRPO's, where GRPO calls more tools than FGPO. Appendix I.3 draws one question from each of two failure pools of FGPO on GenomeQA. The walk-through is drawn from the GenomeQA pool of the second kind, excluding questions already shown. Table 24 gives every pool's size, the failure pools for both policies. Sequences and tool outputs are shown as in Appendix H, with the same two edits; a tool that returned no output is shown as *unavailable*, as the reasoner received it. On GenomeQA's TF-motif family the genomic expert has no reference set and always returns such a line.

Table 24: Pool sizes for the draws of this section. *Subset matters*: some legal subset right and some wrong. *Fewer tools*: right without tools and under FGPO's selection, wrong under GRPO's larger one. *Tools hurt*: right without tools, wrong under the policy's selection. *Missed rescue*: wrong without tools and under the policy's selection, right under some legal subset. Failure pools are given as FGPO / GRPO.

Benchmark	Questions	Subset matters	Fewer tools	Tools hurt	Missed rescue
GenomeQA	3,590	2,237	79	400 / 444	490 / 703
GenBench-X	1,000	807	31	163 / 178	89 / 209
BM4	1,000	802	5	194 / 188	127 / 158

I.1 Where each policy puts its probability

Figure 14 shows, for each drawn question, both converged policies' distributions over the 16 legal subsets. Both policies are concentrated, as the mechanism analysis of Section 3.4 expects of converged policies: FGPO puts at least 0.97 of its legal-string mass on a single subset in every panel, GRPO between 0.38 and 0.82. The panels differ in where that mass lands. On these twelve questions each policy's selection is right on six of them (FGPO 6, GRPO 6); the aggregate difference is the one of Table 2, not something twelve draws can show.

I.2 Fewer tools, right answer

Tables 25–27 show the questions of the *fewer tools* pool, one per task slot where the pool is not empty. In each, the reasoner already answers correctly without tools and keeps the right answer under FGPO's selection, while under GRPO's larger subset it picks a wrong option. The middle column shows what the tools that only GRPO called reported. Where the policy's raw output is not one of the 16 legal strings, it is shown as emitted together with the subset the parser reads from it.

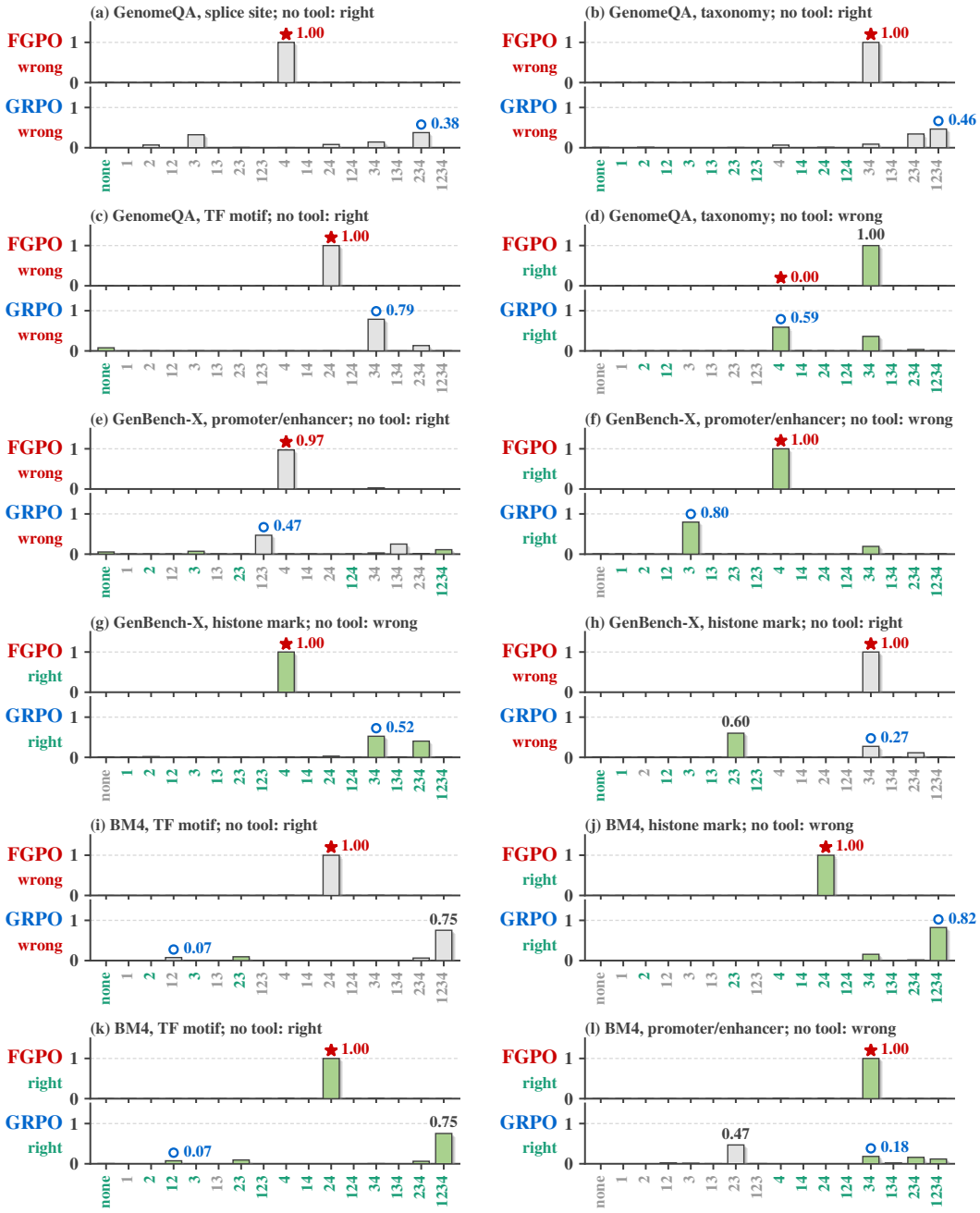


Figure 14: **Each policy's action distribution on twelve drawn questions** (four per benchmark, drawn uniformly from the questions on which the subset matters). Bars are each converged policy's teacher-forced distribution over the 16 legal answer strings (p_θ^{diag}, Appendix F.2); green bars and tick labels mark the subsets under which Qwen3-8B answers correctly, grey the rest. Tick labels list tool indices (34 is T_3+T_4). The star (FGPO) and circle (GRPO) mark the subset parsed from the policy's free-generation output, with its p_θ^{diag} value; the row label says whether that selection is right. A mark on a low bar means greedy decoding did not produce the most probable legal string, or, in panel (d), produced <answer>3,4</answer>, whose out-of-range index the parser drops, leaving $\{T_4\}$ (Appendix B.1).

Table 25: **GenomeQA: fewer tools, right answer**, one per task slot with a non-empty pool, drawn by the rule of Appendix I. The reasoner is right without tools and under FGPO's smaller subset, and wrong under GRPO's larger one. Answers are Qwen3-8B's, right in green and wrong in red; the middle column condenses the output of the tools GRPO added.

No-Tool / FGPO	GRPO: selection and the evidence it added	GRPO
Histone mark (`mcq_4_histone_marks_213`): "Classify the following DNA sequence from K562 cells into the correct histone category: `AGAGAGTATA`...`CATCCTCAGG` (1,000 bp)" Options: A. H2AFZ; B. H3K9ac; C. H3K4me1; D. H3K27ac. [GT: A]		
No-Tool: A FGPO emits `<answer>3,4</answer>`, i.e. T_4: A	GRPO emits `<answer>1,2,3</answer>`, i.e. T_2+T_3+T_4. Output of the tools only GRPO called: T_2 motif hits: NFKB1, GABPA, MYC, SPI1, MEF2A, NFYA, USF1, CTCF, ESR1 T_3 splice pattern: no site at the candidate position, only background hits elsewhere	D
Promoter/enhancer (`mcq_1_enhancer_vs_promoter_105`): "Please filter through these options. Which sequence holds a promoter instead of the enhancer found in the other three?" Options: four 400-bp sequences. [GT: D]		
No-Tool: D FGPO emits `<answer>3</answer>`, i.e. T_4: D	GRPO emits `<answer>0,1,2,3</answer>`, i.e. T_1+T_2+T_3+T_4. Output of the tools only GRPO called: T_1 composition: A: GC 0.43, CpG o/e 0.11 | B: GC 0.58, CpG o/e 0.30 | C: GC 0.45, CpG o/e 0.32 | D: GC 0.48, CpG o/e 0.89 T_2 motif hits: A: FOXA1, MEF2A, RUNX1, GATA1 | B: SPI1 | C: TEAD4 | D: STAT1, SPI1, MAX, MYC, GABPA, USF1, FOXA1 T_3 splice pattern: all options: no site at the candidate position, only background hits elsewhere	A
Splice site (`mcq_2_splice_sites_340`): "What kind of splice sites are present in this Human DNA sequence? Sequence: `GACCTACACA`...`CTTTAAGGAT` (600 bp)" Options: A. Neither (No Splice Site); B. Both Acceptor and Donor; C. Only Donor; D. Only Acceptor. [GT: B]		
No-Tool: B FGPO emits `<answer>2,3</answer>`, i.e. T_3+T_4: B	GRPO emits `<answer>0,1,2,3</answer>`, i.e. T_1+T_2+T_3+T_4. Output of the tools only GRPO called: T_1 composition: GC 0.39, CpG o/e 0.17 T_2 motif hits: CEBPB	C
Taxonomy (`mcq_3_species_cls_167`): "Select the DNA sequence that is derived from a Eukaryote genome." Options: four 1000-bp sequences. [GT: B]		
No-Tool: B FGPO emits `<answer>3</answer>`, i.e. T_4: B	GRPO emits `<answer>0,1,2,3</answer>`, i.e. T_1+T_2+T_3+T_4. Output of the tools only GRPO called: T_1 composition: A: GC 0.67, CpG o/e 1.18 | B: GC 0.63, CpG o/e 0.24 | C: GC 0.38, CpG o/e 1.13 | D: GC 0.62, CpG o/e 1.14 T_2 motif hits: A: MAX, MYC, NRF1, E2F1, YY1, EGR1 | B: USF1, REST, NRF1, SP1, SPI1, GABPA | C: FOS, MEF2A, YY1, PAX5, JUN, FOXA1 | D: ZBTB33, EGR1, REST, SPI1 T_3 splice pattern: all options: no site at the candidate position, only background hits elsewhere	A
TF motif (`mcq_5_tfbs_494`): "Which of the following Human DNA sequences contains a binding site for NRF1?" Options: four 100-bp sequences. [GT: B]		
No-Tool: B FGPO emits `<answer>1</answer>`, i.e. T_2: B	GRPO emits `<answer>0,1,2,3</answer>`, i.e. T_1+T_2+T_3+T_4. Output of the tools only GRPO called: T_1 composition: A: GC 0.42, CpG o/e 0.23 | B: GC 0.69, CpG o/e 1.02 | C: GC 0.42, CpG o/e 0.00 | D: GC 0.57, CpG o/e 0.25 T_3 splice pattern: A: no site at the candidate position, only background hits elsewhere | B: no splice signal anywhere in the window | C: acceptor at the candidate position | D: no splice signal anywhere in the window T_4 kNN prediction: all options: unavailable: no reference set for this task	A

Table 26: **GenBench-X: fewer tools, right answer**, one per task slot with a non-empty pool, drawn by the rule of Appendix I. The reasoner is right without tools and under FGPO's smaller subset, and wrong under GRPO's larger one. Answers are Qwen3-8B's, right in green and wrong in red; the middle column condenses the output of the tools GRPO added.

No-Tool / FGPO	GRPO: selection and the evidence it added	GRPO
`human_ocr_ensembl` (histone-mark slot, `gb2_human_ocr_en_sel_0043`): "Which of the following Human DNA sequences lies in an open chromatin region?" Options: four sequences (374, 250, 245, 289 bp). [GT: D]		
No-Tool: D FGPO emits `<answer>3</answer>`, i.e. T_4: D	GRPO emits `<answer>1,2,3</answer>`, i.e. T_2+T_3+T_4. Output of the tools only GRPO called: T_2 motif hits: A: GATA1, GABPA, SRF | B: MEF2A | C: MEF2A, SPI1 | D: RUNX1, YY1 T_3 splice pattern: all options: no site at the candidate position, only background hits elsewhere	A
`human_ensembl_regulatory` (promoter/enhancer slot, `gb2_human_ensemb_0_0028`): "Which regulatory element does this Human DNA sequence represent?" Options: A. Enhancer; B. Open chromatin region; C. Promoter. [GT: A]		
No-Tool: A FGPO emits `<answer>3</answer>`, i.e. T_4: A	GRPO emits `<answer>0,1,2</answer>`, i.e. T_1+T_2+T_3. Output of the tools only GRPO called: T_1 composition: GC 0.37, CpG o/e 0.15 T_2 motif hits: TEAD4, STAT1 T_3 splice pattern: donor at the candidate position	C
`demo_coding_vs_intergenomic` (splice-site slot, `gb2_demo_coding__sel_0050`): "Which of the following DNA sequences is a Intergenomic sequence?" Options: four 200-bp sequences. [GT: C]		
No-Tool: C FGPO emits `<answer>3</answer>`, i.e. T_4: C	GRPO emits `<answer>2,3</answer>`, i.e. T_3+T_4. Output of the tools only GRPO called: T_3 splice pattern: A: no splice signal anywhere in the window | B: no splice signal anywhere in the window | C: no site at the candidate position, only background hits elsewhere | D: no splice signal anywhere in the window	A
`demo_human_or_worm` (taxonomy slot, `gb2_demo_human_o_sel_0016`): "Which of the following DNA sequences is from Human?" Options: four 200-bp sequences. [GT: D]		
No-Tool: D FGPO emits `<answer>3</answer>`, i.e. T_4: D	GRPO emits `<answer>0,1,2,3</answer>`, i.e. T_1+T_2+T_3+T_4. Output of the tools only GRPO called: T_1 composition: A: GC 0.29, CpG o/e 0.69 | B: GC 0.23, CpG o/e 1.58 | C: GC 0.28, CpG o/e 0.49 | D: GC 0.45, CpG o/e 0.10 T_2 motif hits: A: TP53 | B: no hit | C: PAX5 | D: MEF2A, YY1 T_3 splice pattern: A: no site at the candidate position, only background hits elsewhere | B: no splice signal anywhere in the window | C: no site at the candidate position, only background hits elsewhere | D: no splice signal anywhere in the window	A

Table 27: **BM4: fewer tools, right answer**, one per task slot with a non-empty pool, drawn by the rule of Appendix I. The reasoner is right without tools and under FGPO's smaller subset, and wrong under GRPO's larger one. Answers are Qwen3-8B's, right in green and wrong in red; the middle column condenses the output of the tools GRPO added.

No-Tool / FGPO	GRPO: selection and the evidence it added	GRPO
Bacterial σ^{70} promoters (promoter/enhancer slot, `bm4_sigma70_sel_0075`): "Which of the following Escherichia coli DNA sequences is a Promoter?" Options: four 81-bp sequences. [GT: B]		
No-Tool: B FGPO emits `<answer>1</answer>`, i.e. T_2: B	GRPO emits `<answer>0,1,2,3</answer>`, i.e. T_1+T_2+T_3+T_4. Output of the tools only GRPO called: T_1 composition: A: GC 0.31, CpG o/e 1.04 | B: GC 0.30, CpG o/e 1.70 | C: GC 0.37, CpG o/e 0.38 | D: GC 0.38, CpG o/e 0.35 T_3 splice pattern: A: no splice signal anywhere in the window | B: no splice signal anywhere in the window | C: no splice signal anywhere in the window | D: acceptor at the candidate position T_4 kNN prediction: A: Promoter (0.55) | B: Promoter (0.60) | C: Promoter (0.60) | D: Promoter (0.60)	A
Bacteria vs. archaea (taxonomy slot, `bm4_domain_sel_0040`): "Which of the following DNA sequences is from Bacteria?" Options: four 81-bp sequences. [GT: D]		
No-Tool: D FGPO emits `<answer>3</answer>`, i.e. T_4: D	GRPO emits `<answer>0,2,3</answer>`, i.e. T_1+T_3+T_4. Output of the tools only GRPO called: T_1 composition: A: GC 0.60, CpG o/e 1.49 | B: GC 0.42, CpG o/e 1.41 | C: GC 0.35, CpG o/e 0.84 | D: GC 0.74, CpG o/e 1.08 T_3 splice pattern: A: no splice signal anywhere in the window | B: no site at the candidate position, only background hits elsewhere | C: no splice signal anywhere in the window | D: no splice signal anywhere in the window	A

I.3 Failures of FGPO

FGPO also fails. Table 24 gives its failure pools next to GRPO's: they are smaller on every benchmark except the BM4 *tools hurt* pool (194 against 188). Table 28 shows one question from each GenomeQA pool. In the *tools hurt* case only the empty subset is right, so every tool call, GRPO's included, loses the answer; in the *missed rescue* a different subset would have been right, and the last line of the middle column lists which.

Table 28: **Failures of FGPO** on GenomeQA, drawn by the rule of Appendix I: one where FGPO's tools turn a right no-tool answer wrong (*tools hurt*) and one where FGPO is wrong although some subset would have been right (*missed rescue*). Subsets are written by tool index (1+3 is T_1+T_3).

Kind / No-Tool / GRPO	FGPO: selection and tool evidence	FGPO
Taxonomy (`mcq_3_species_cls_411`): "Select the sequence that does not belong to the Virus group, as it is essentially a Prokaryote sequence." Options: four 1000-bp sequences. [GT: C]		
tools hurt No-Tool: C GRPO (T_2+T_3+T_4): A	FGPO emits `<answer>3</answer>`, i.e. T_4. T_4 kNN prediction: A: Virus (0.41) | B: Eukaryote (0.47) | C: Eukaryote (0.44) | D: Virus (0.45) Subsets under which the reasoner is right: no tool.	A
Splice site (`mcq_2_splice_sites_168`): "Identify the sequence that does NOT harbor a valid biological splice site motif." Options: four 600-bp sequences. [GT: A]		
missed rescue No-Tool: D GRPO (T_3+T_4): A	FGPO emits `<answer>3</answer>`, i.e. T_4. T_4 kNN prediction: A: Only Acceptor (0.35) | B: Neither (No Splice Site) (0.44) | C: Neither (No Splice Site) (0.39) | D: Only Donor (0.32) Subsets under which the reasoner is right: 1, 2, 1+2, 1+3, 2+3, 1+2+3, 1+4, 2+4, 1+2+4, 3+4, 1+3+4, 2+3+4, 1+2+3+4.	B

I.4 Walk-through: fewer tools (GenomeQA, promoter/enhancer)

Drawn from the GenomeQA *fewer tools* pool (Appendix I.2).

Question (`mcq_1_enhancer_vs_promoter_222`). "Please filter through these options. Which sequence holds a promoter instead of the enhancer found in the other three?"

Option A (400 bp):
CCACCAGGCACACAAGGACCCCAGAAGGAGAAGAGACCCCACCCCGGGGGGTCCCTGGAG
GAATGGGAAGCCCCCACGGAGAATGGGGGCCAGGAGCTGACATGGGGGGGAGCAGGGAGG
GGGAGGCTCCCATCTTCCTTGAGAGGAGAGAGGGCTGTGGCCCCACCTCTGGCTCCAAAC
CCAGCCCAGCCCCTGGCATCCTGCCTTCCAGGAAGCCTCCGGCCCGGGGCAGGGAGGGGT
GGGCAGGATGCTGCGTGGCAGAGGTGGGGAGAAACAGTCAACCCGCAGCTGACCCTCCTG
CCCAGTGCCTGGTCCACGACCTGCATTCACACTCCCGGGCCTGGCCAAGGCTGCCTGGAT
GCGGCCGCGGCCCCAGCAGGGCCACTCCCTGGCAGGCCGC
Option B (400 bp):
GGCCCGGGGCGCCCTCCGGGGGCCGGCACACAGCAGGGTCTCGCTTCCTCGCGCTGTCTT
CTTCAGATGGGATCACTTCGACTTGCGGCCGTTTGGGCGTTCGCGGGGCCTCGGCGATTC
TGTGGCGTCCTGGGCTGTGTATGTAGCCCCAGCGGCCGAGGCTCACGGATCCGACCGCCC
GGCCGCCGCGTCTGTCCCAAGTTGGGAGACGGGAGGGCCGGGTGGGGTCGCGCCTGCGCC
AGGGGGCGGCTCAGAACCAGACTCGCGGTTTTTAACTTATTCCTCAGAGGGGCGAAAGCG
AGTCTCGACAAGGACGGCGACGGCTCTCTAGAAGCCGGGGTCCAGGTAGGCTGCCCGGCC
CTCGAGCTCGCCCAGCGCCGCTCATGATGCGCCCCGCAGC
Option C (400 bp):
CTTCCTCCGTTGTCCCTCCATGCTGCGGCCATCTTTTCAGCAAAGGCATCTGGGCTTTTT
CCTGTCGGTCAGCGGGTGACTCCTTCCTCCGTCGTCCCTCCATGCGTGCGGCCATCTTTT
CAGGAAAGGCATCTGTGCTTTTTCCTGTTGGTCAGCGGGTGACTCCTTCCTCCGTCGTCC
CACCATGCGTGCGGCCATCTTTTGTTGCGTGCCTACACTGTCGATGTCAGGCTGGGGACA
CACAGGACCGACCGAGCATGGTGAGATCTAGAATTTATGCCTCATTCAGTTAATGTTCTT
TAGATATTTGTTCAACATTTATTTGGTAAGAAGTGGGTAAAAGTTTCCACTGATAAAGAC
CTTACACTTTCTCAGTAACCTATGTACCAGCTGTCCTTTC
Option D (400 bp):
TGGGAGCGGAGCTTGCAGTGAGCCGAGATCGCACCACTGCACTCCATCCTGGGGGACAGA
GAGAGACTCTGTCTCAAAAAAAAAAGACTCCAAGGGAAAGCTGCCTGGGTCGTGACCTGG
CACGTCCAAGTCGCTGCTTAGGCTTGCAAGGGGGTCCAGACCAGGCTGTGGATGGGAAAT
GGGAGGCTGGGGGAGGATGGAGGCTGAGCAGTGGCTGGATGTTGGGGTGCCCTGTTGAGG
GGCAGCTGGCAGCAGCCACGGACACAGGGGTGGGTGCAAAGACACTGAGTTTCTGTTCCA
GGAGGAAGGGGGGCCAGCTGGCAGAGGAGAGATCACCCTGGGCAATTTGGTGGTGCCTGG
GGTGCACACTTCCCCAGAGGCAGCAGGAGGAGGAGCTGGA

Outcome. Ground truth: B. Without tools Qwen3-8B answers B. GRPO emits `<answer>0,1,2,3</answer>`, i.e. T_1+T_2+T_3+T_4, and the reasoner answers A. FGPO emits `<answer>3</answer>`, i.e. T_4, and the reasoner answers B. The table is this question's row of the exhaustive reward table, with FGPO's choice marked * and GRPO's $^\circ$; – marks a reply with no parseable letter.

Subset	none	1	2	1+2	3	1+3	2+3	1+2+3
Answer	B	A	A	A	A	A	A	A
Subset	4*	1+4	2+4	1+2+4	3+4	1+3+4	2+3+4	1+2+3+4°
Answer	B	A	B	A	A	A	A	A

Tool output passed to the reasoner, for every tool either policy selected, with the policies that selected it.

Option A:
`[T1] sequence_statistics:` *(selected by GRPO)*


```
{
  "length_bp": 400,
  "gc_content": 0.695,
  "at_content": 0.305,
  "cpg_density": 0.03008,
  "cpg_observed_over_expected": 0.2486,
  "gc_skew": 0.0288,
  "at_skew": 0.2623,
  "base_entropy_bits": 1.8716,
  "n_fraction": 0.0,
  "top_3mers": [
    {"kmer": "GGG", "count": 28, "frequency": 0.07035},
    {"kmer": "CCC", "count": 25, "frequency": 0.06281},
    {"kmer": "GGC", "count": 19, "frequency": 0.04774},
    {"kmer": "AGG", "count": 18, "frequency": 0.04523},
```

```
      {"kmer": "CCA", "count": 17, "frequency": 0.04271}
    ],
    "trimer_entropy_bits": 5.2387,
    "top_4mers": [
      {"kmer": "GGGG", "count": 14, "frequency": 0.03526},
      {"kmer": "GGAG", "count": 10, "frequency": 0.02519},
      {"kmer": "CCCA", "count": 9, "frequency": 0.02267},
      {"kmer": "CCTG", "count": 9, "frequency": 0.02267},
      {"kmer": "GGCC", "count": 9, "frequency": 0.02267}
    ],
    "top_6mers": [
      {"kmer": "GGAGAA", "count": 3, "frequency": 0.00759},
      {"kmer": "CCCCAC", "count": 3, "frequency": 0.00759},
      {"kmer": "CCCGGG", "count": 3, "frequency": 0.00759},
      {"kmer": "GGGGGG", "count": 3, "frequency": 0.00759},
      {"kmer": "CCCTGG", "count": 3, "frequency": 0.00759}
    ],
    "gc_profile": {"window_bp": 40, "max_window_gc": 0.85, "min_window_gc": 0.6,
        "gc_std_across_windows": 0.0543},
    "longest_homopolymer_bp": 7,
    "unique_4mer_coverage": 0.5273,
    "interpretation": {"cpg_island": "no - GC 0.69, CpG observed/expected 0.25; a CpG island needs
        GC>0.50 and O/E>0.60", "gc_level": "high, typical of CpG islands and promoter regions", "note":
        "CpG islands mark promoters; enhancers are usually not CpG islands."}
}
```


[T2] motif_scanner: *(selected by GRPO)*


```
{
  "backend": "biopython",
  "database": "JASPAR2024",
  "factors_named_in_question": null,
  "n_motifs_scanned": 32,
  "p_value_threshold": 0.0001,
  "n_significant_hits": 5,
  "matched_transcription_factors": ["STAT1", "RELA", "SP1", "NFKB1"],
  "hits_per_factor": {"STAT1": 2, "RELA": 1, "SP1": 1, "NFKB1": 1},
  "hits": [
    {"motif": "STAT1", "position": 205, "strand": "-", "score": 12.927, "relative_score": 0.823,
        "p_value": 1.907e-05, "matched_sequence": "TTCCAGGAA"},
    {"motif": "RELA", "position": 64, "strand": "-", "score": 11.631, "relative_score": 0.693,
        "p_value": 4.196e-05, "matched_sequence": "GGGAAGCCCC"},
    {"motif": "SP1", "position": 106, "strand": "+", "score": 10.217, "relative_score": 0.621,
        "p_value": 7.629e-05, "matched_sequence": "GGGGAGCAG"},
    {"motif": "NFKB1", "position": 62, "strand": "-", "score": 2.064, "relative_score": 0.088,
        "p_value": 9.502e-05, "matched_sequence": "ATGGGAAGCCCCC"},
    {"motif": "STAT1", "position": 205, "strand": "+", "score": 15.707, "relative_score": 1.0,
        "p_value": 0.0001111, "matched_sequence": "TTCCAGGAA"}
  ]
}
```


[T3] splice_analyzer: *(selected by GRPO)*


```
{
  "method": "MaxEntScan",
  "candidate_position": 200,
  "centre_window_bp": 10,
  "donor_threshold_bits": 6.0,
  "acceptor_threshold_bits": 6.0,
  "donor_at_candidate_position": {"found": false},
  "acceptor_at_candidate_position": {"found": true, "position": 209, "score_bits": 8.789, "context":
      "CCTGGCATCCTGCCTTCCAGGAA"},
  "overall_pattern": "acceptor at the candidate position",
  "best_donor_anywhere": null,
  "best_acceptor_anywhere": {"position": 209, "score_bits": 8.789, "context":
      "CCTGGCATCCTGCCTTCCAGGAA"},
  "background_donor_hits": 0,
  "background_acceptor_hits": 0,
  "interpretation": "A strong MaxEntScan score at the candidate position is the evidence that matters;
      hits elsewhere in the window are background. Donor scores near 8-11 bits are strong, 6-8
      marginal."
}
```


[T4] genomic_expert: *(selected by FGPO and GRPO)*


```
{
  "backbone": "[path redacted]",
  "backbone_frozen": true,
  "method": "cosine kNN (k=15) over frozen embeddings, self-hits excluded",
  "task": "promoter_enhancer",
  "prediction": "Enhancer",
  "confidence": 0.4759,
  "class_probabilities": {"Enhancer": 0.4759, "Strong enhancer": 0.1816, "Promoter": 0.1598, "Weak
      enhancer": 0.1246, "Not an enhancer": 0.0581},
  "nearest_neighbour_labels": ["Enhancer", "Enhancer", "Promoter", "Promoter", "Enhancer", "Weak
      enhancer", "Strong enhancer", "Enhancer", "Strong enhancer", "Enhancer"],
  "mean_neighbour_distance": 0.0138,
```

```
  "n_self_hits_excluded":  0
}
```


Option B:

[T1] sequence_statistics: *(selected by GRPO)*


```
{
  "length_bp":  400,
  "gc_content":  0.695,
  "at_content":  0.305,
  "cpg_density":  0.11779,
  "cpg_observed_over_expected":  0.9749,
  "gc_skew":  0.0432,
  "at_skew":  -0.1475,
  "base_entropy_bits":  1.8816,
  "n_fraction":  0.0,
  "top_3mers":  [
    {"kmer":  "GGG", "count":  24, "frequency":  0.0603},
    {"kmer":  "GGC", "count":  21, "frequency":  0.05276},
    {"kmer":  "GCC", "count":  19, "frequency":  0.04774},
    {"kmer":  "GCG", "count":  18, "frequency":  0.04523},
    {"kmer":  "CGG", "count":  17, "frequency":  0.04271}
  ],
  "trimer_entropy_bits":  5.5257,
  "top_4mers":  [
    {"kmer":  "GGGG", "count":  9, "frequency":  0.02267},
    {"kmer":  "CGGC", "count":  9, "frequency":  0.02267},
    {"kmer":  "GGCC", "count":  8, "frequency":  0.02015},
    {"kmer":  "GCCC", "count":  8, "frequency":  0.02015},
    {"kmer":  "GGGC", "count":  8, "frequency":  0.02015}
  ],
  "top_6mers":  [
    {"kmer":  "GCCCGG", "count":  3, "frequency":  0.00759},
    {"kmer":  "CCGGGG", "count":  3, "frequency":  0.00759},
    {"kmer":  "GGGGCG", "count":  3, "frequency":  0.00759},
    {"kmer":  "CGGCCG", "count":  3, "frequency":  0.00759},
    {"kmer":  "CGGGGC", "count":  2, "frequency":  0.00506}
  ],
  "gc_profile":  {"window_bp":  40, "max_window_gc":  0.875, "min_window_gc":  0.45,
      "gc_std_across_windows":  0.0987},
  "longest_homopolymer_bp":  5,
  "unique_4mer_coverage":  0.668,
  "interpretation":  {"cpg_island":  "yes - GC 0.69 and CpG observed/expected 0.97 both clear the
      standard thresholds (GC>0.50, O/E>0.60 over >=200 bp)", "gc_level":  "high, typical of CpG
      islands and promoter regions", "note":  "CpG islands mark promoters; enhancers are usually not
      CpG islands."}
}
```


[T2] motif_scanner: *(selected by GRPO)*


```
{
  "backend":  "biopython",
  "database":  "JASPAR2024",
  "factors_named_in_question":  null,
  "n_motifs_scanned":  32,
  "p_value_threshold":  0.0001,
  "n_significant_hits":  4,
  "matched_transcription_factors":  ["PAX5", "NRF1", "CTCF"],
  "hits_per_factor":  {"NRF1":  1, "PAX5":  2, "CTCF":  1},
  "hits":  [
    {"motif":  "NRF1", "position":  227, "strand":  "+", "score":  14.735, "relative_score":  0.734,
        "p_value":  4.53e-06, "matched_sequence":  "TCGCGCCTGCGC"},
    {"motif":  "PAX5", "position":  36, "strand":  "-", "score":  10.818, "relative_score":  0.747,
        "p_value":  7.629e-05, "matched_sequence":  "GGTCTCGC"},
    {"motif":  "CTCF", "position":  8, "strand":  "-", "score":  10.739, "relative_score":  0.658,
        "p_value":  8.583e-05, "matched_sequence":  "GCGCCCTCCG"},
    {"motif":  "PAX5", "position":  225, "strand":  "-", "score":  10.724, "relative_score":  0.74,
        "p_value":  9.155e-05, "matched_sequence":  "GGTCGCGC"}
  ]
}
```


[T3] splice_analyzer: *(selected by GRPO)*


```
{
  "method":  "MaxEntScan",
  "candidate_position":  200,
  "centre_window_bp":  10,
  "donor_threshold_bits":  6.0,
  "acceptor_threshold_bits":  6.0,
  "donor_at_candidate_position":  {"found":  false},
  "acceptor_at_candidate_position":  {"found":  false},
  "overall_pattern":  "no site at the candidate position, only background hits elsewhere",
  "best_donor_anywhere":  {"position":  345, "score_bits":  9.878, "context":  "CAGGTAGGC"},
  "best_acceptor_anywhere":  {"position":  64, "score_bits":  11.107, "context":
      "CCTCGCGCTGTCTTCTTCAGATG"},
  "background_donor_hits":  1,
  "background_acceptor_hits":  2,
```

```
  "interpretation": "Nothing scores above threshold at the candidate position, which argues against a
      functional splice site here."
}
[T4] genomic_expert: (selected by FGPO and GRPO)
{
  "backbone": "[path redacted]",
  "backbone_frozen": true,
  "method": "cosine kNN (k=15) over frozen embeddings, self-hits excluded",
  "task": "promoter_enhancer",
  "prediction": "Promoter",
  "confidence": 0.4489,
  "class_probabilities": {"Promoter": 0.4489, "Enhancer": 0.2617, "Weak enhancer": 0.2274, "Strong
      enhancer": 0.0619},
  "nearest_neighbour_labels": ["Promoter", "Promoter", "Enhancer", "Promoter", "Promoter", "Strong
      enhancer", "Enhancer", "Weak enhancer", "Promoter", "Promoter"],
  "mean_neighbour_distance": 0.0152,
  "n_self_hits_excluded": 0
}
```


Option C:


```
[T1] sequence_statistics: (selected by GRPO)
{
  "length_bp": 400,
  "gc_content": 0.5,
  "at_content": 0.5,
  "cpg_density": 0.04261,
  "cpg_observed_over_expected": 0.6869,
  "gc_skew": -0.1,
  "at_skew": -0.28,
  "base_entropy_bits": 1.9677,
  "n_fraction": 0.0,
  "top_3mers": [
    {"kmer": "TTT", "count": 20, "frequency": 0.05025},
    {"kmer": "TCC", "count": 17, "frequency": 0.04271},
    {"kmer": "CCT", "count": 14, "frequency": 0.03518},
    {"kmer": "TTC", "count": 13, "frequency": 0.03266},
    {"kmer": "CTT", "count": 12, "frequency": 0.03015}
  ],
  "trimer_entropy_bits": 5.7505,
  "top_4mers": [
    {"kmer": "TCCT", "count": 8, "frequency": 0.02015},
    {"kmer": "CTTT", "count": 8, "frequency": 0.02015},
    {"kmer": "CTCC", "count": 7, "frequency": 0.01763},
    {"kmer": "TTTT", "count": 7, "frequency": 0.01763},
    {"kmer": "TTTC", "count": 7, "frequency": 0.01763}
  ],
  "top_6mers": [
    {"kmer": "CTTCCT", "count": 3, "frequency": 0.00759},
    {"kmer": "TTCCTC", "count": 3, "frequency": 0.00759},
    {"kmer": "TCCTCC", "count": 3, "frequency": 0.00759},
    {"kmer": "CCTCCG", "count": 3, "frequency": 0.00759},
    {"kmer": "CTCCGT", "count": 3, "frequency": 0.00759}
  ],
  "gc_profile": {"window_bp": 40, "max_window_gc": 0.7, "min_window_gc": 0.2,
      "gc_std_across_windows": 0.1313},
  "longest_homopolymer_bp": 5,
  "unique_4mer_coverage": 0.6719,
  "interpretation": {"cpg_island": "no - GC 0.50, CpG observed/expected 0.69; a CpG island needs
      GC>0.50 and O/E>0.60", "gc_level": "moderate", "note": "CpG islands mark promoters; enhancers
      are usually not CpG islands."}
}
[T2] motif_scanner: (selected by GRPO)
{
  "backend": "biopython",
  "database": "JASPAR2024",
  "factors_named_in_question": null,
  "n_motifs_scanned": 32,
  "p_value_threshold": 0.0001,
  "n_significant_hits": 0,
  "matched_transcription_factors": [],
  "hits_per_factor": {},
  "hits": []
}
[T3] splice_analyzer: (selected by GRPO)
{
  "method": "MaxEntScan",
  "candidate_position": 200,
  "centre_window_bp": 10,
  "donor_threshold_bits": 6.0,
  "acceptor_threshold_bits": 6.0,
  "donor_at_candidate_position": {"found": false},
  "acceptor_at_candidate_position": {"found": false},
```

```
  "overall_pattern": "no site at the candidate position, only background hits elsewhere",
  "best_donor_anywhere": {"position": 325, "score_bits": 8.846, "context": "TTGGTAAGA"},
  "best_acceptor_anywhere": {"position": 121, "score_bits": 7.861, "context":
      "GCGTGCGGCCATCTTTTCAGGAA"},
  "background_donor_hits": 2,
  "background_acceptor_hits": 2,
  "interpretation": "Nothing scores above threshold at the candidate position, which argues against a
      functional splice site here."
}
```


[T4] genomic_expert: *(selected by FGPO and GRPO)*


```
{
  "backbone": "[path redacted]",
  "backbone_frozen": true,
  "method": "cosine kNN (k=15) over frozen embeddings, self-hits excluded",
  "task": "promoter_enhancer",
  "prediction": "Promoter",
  "confidence": 0.3324,
  "class_probabilities": {"Promoter": 0.3324, "Enhancer": 0.2884, "Not an enhancer": 0.1844, "Strong
      enhancer": 0.1292, "Weak enhancer": 0.0656},
  "nearest_neighbour_labels": ["Enhancer", "Promoter", "Promoter", "Promoter", "Strong enhancer", "Weak
      enhancer", "Enhancer", "Enhancer", "Not an enhancer", "Strong enhancer"],
  "mean_neighbour_distance": 0.0136,
  "n_self_hits_excluded": 0
}
```


Option D:

[T1] sequence_statistics: *(selected by GRPO)*


```
{
  "length_bp": 400,
  "gc_content": 0.61,
  "at_content": 0.39,
  "cpg_density": 0.01754,
  "cpg_observed_over_expected": 0.205,
  "gc_skew": 0.2869,
  "at_skew": 0.1538,
  "base_entropy_bits": 1.9214,
  "n_fraction": 0.0,
  "top_3mers": [
    {"kmer": "GGG", "count": 28, "frequency": 0.07035},
    {"kmer": "GAG", "count": 21, "frequency": 0.05276},
    {"kmer": "TGG", "count": 20, "frequency": 0.05025},
    {"kmer": "GGA", "count": 19, "frequency": 0.04774},
    {"kmer": "CTG", "count": 19, "frequency": 0.04774}
  ],
  "trimer_entropy_bits": 5.3623,
  "top_4mers": [
    {"kmer": "GGGG", "count": 13, "frequency": 0.03275},
    {"kmer": "TGGG", "count": 10, "frequency": 0.02519},
    {"kmer": "GGAG", "count": 10, "frequency": 0.02519},
    {"kmer": "CTGG", "count": 10, "frequency": 0.02519},
    {"kmer": "GCTG", "count": 9, "frequency": 0.02267}
  ],
  "top_6mers": [
    {"kmer": "AAAAAA", "count": 5, "frequency": 0.01266},
    {"kmer": "CCTGGG", "count": 4, "frequency": 0.01013},
    {"kmer": "GGAGGA", "count": 4, "frequency": 0.01013},
    {"kmer": "CTGGGG", "count": 3, "frequency": 0.00759},
    {"kmer": "GAGAGA", "count": 3, "frequency": 0.00759}
  ],
  "gc_profile": {"window_bp": 40, "max_window_gc": 0.725, "min_window_gc": 0.375,
      "gc_std_across_windows": 0.0699},
  "longest_homopolymer_bp": 10,
  "unique_4mer_coverage": 0.5781,
  "interpretation": {"cpg_island": "no - GC 0.61, CpG observed/expected 0.20; a CpG island needs
      GC>0.50 and O/E>0.60", "gc_level": "high, typical of CpG islands and promoter regions", "note":
      "CpG islands mark promoters; enhancers are usually not CpG islands."}
}
```


[T2] motif_scanner: *(selected by GRPO)*


```
{
  "backend": "biopython",
  "database": "JASPAR2024",
  "factors_named_in_question": null,
  "n_motifs_scanned": 32,
  "p_value_threshold": 0.0001,
  "n_significant_hits": 1,
  "matched_transcription_factors": ["SPI1"],
  "hits_per_factor": {"SPI1": 1},
  "hits": [
    {"motif": "SPI1", "position": 298, "strand": "+", "score": 11.663, "relative_score": 0.533,
        "p_value": 3.156e-05, "matched_sequence": "CAGGAGGAAGGGG"}
  ]
}
```

[T3] splice_analyzer: *(selected by GRPO)*


```
{
  "method": "MaxEntScan",
  "candidate_position": 200,
  "centre_window_bp": 10,
  "donor_threshold_bits": 6.0,
  "acceptor_threshold_bits": 6.0,
  "donor_at_candidate_position": {"found": false},
  "acceptor_at_candidate_position": {"found": false},
  "overall_pattern": "no site at the candidate position, only background hits elsewhere",
  "best_donor_anywhere": null,
  "best_acceptor_anywhere": {"position": 299, "score_bits": 7.83, "context":
      "ACACTGAGTTTCTGTTCCAGGAG"},
  "background_donor_hits": 0,
  "background_acceptor_hits": 1,
  "interpretation": "Nothing scores above threshold at the candidate position, which argues against a
      functional splice site here."
}
```


[T4] genomic_expert: *(selected by FGPO and GRPO)*


```
{
  "backbone": "[path redacted]",
  "backbone_frozen": true,
  "method": "cosine kNN (k=15) over frozen embeddings, self-hits excluded",
  "task": "promoter_enhancer",
  "prediction": "Enhancer",
  "confidence": 0.6305,
  "class_probabilities": {"Enhancer": 0.6305, "Promoter": 0.1903, "Strong enhancer": 0.1193, "Weak
      enhancer": 0.0599},
  "nearest_neighbour_labels": ["Enhancer", "Enhancer", "Enhancer", "Enhancer", "Promoter", "Enhancer",
      "Enhancer", "Enhancer", "Promoter", "Enhancer"],
  "mean_neighbour_distance": 0.0185,
  "n_self_hits_excluded": 0
}
```